DLR-IB-AS-BS-2020-20

Mehrgitteruntersuchungen zur Lösung der Navier-Stokes-Gleichungen

Forschungsbericht

Autor
Sophie Pientka

DLR Deutsches Zentrum für Luft- und Raumfahrt

Bericht des Instituts für Aerodynamik und Strömungstechnik
Report of the Institute of Aerodynamics and Flow Technology

DLR-IB-AS-BS-2020-20

Mehrgitteruntersuchungen zur Lösung der Navier-Stokes-Gleichungen

Sophie Pientka

Herausgeber:
Deutsches Zentrum für Luft- und Raumfahrt e.V.
Institut für Aerodynamik und Strömungstechnik
Lilienthalplatz 7, 38108 Braunschweig

ISSN 1614-7790

Stufe der Zugänglichkeit: 1
Braunschweig, im Februar 2020

Institutsdirektor:
Prof. Dr.-Ing. habil. C.-C. Rossow

Verfasser:
Sophie Pientka

Abteilung: Center of Computer Applications in Aerospace Science and Engineering

Abteilungsleiter:
Prof. Dr. S. Görtz

Der Bericht enthält:
102 Seiten
36 Bilder
5 Tabellen
43 Literaturstellen

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

Kapitel 1

Einleitung

Der Luftverkehr ist sowohl in Deutschland als auch international der am stärksten wachsende Verkehrsträger [4]. Aus diesem Grund ist es von besonders großer Bedeutung, die Effizienz von Flugzeugen zu verbessern. Neben dem klassischen Testen neuer Bauteile im Windkanal wird heutzutage oft auf die numerische Simulation, auch CFD (computational fluid dynamics) genannt, zurückgegriffen. Dabei werden Probleme der Strömungsdynamik, beispielsweise an Flugzeugflügeln, mithilfe von numerischer Methoden approximativ gelöst.
Die Abteilung $C^2A^2S^2E$ (Center for Computer Applications in AeroSpace Science and Engineering) im Institut für Aerodynamik und Strömungstechnik des Deutschen Zentrums für Luft- und Raumfahrt (DLR) beschäftigt sich mit der Entwicklung numerischer Methoden zur Simulation und Optimierung von Luftfahrzeugen [22]. Die dabei für die Simulation verwendete Software ist der sogenannte TAU-Code [26] [27]. Dies ist eine vom DLR entwickelte Software zum numerischen Lösen der Euler- bzw. Navier-Stokes-Gleichungen auf einem unstrukturierten Gitter.
Das zugrundeliegende mathematische Modell zur Simulation von Strömungen sind die Navier-Stokes-Gleichungen und im reibungsfreien Fall die Euler-Gleichungen. Um die Strömungsdynamik eines Flugzeugflügels zu simulieren, wird zunächst ein Gebiet um einen Teil eines Flugzeuges (z.B. einen Flügel) oder um ein gesamtes Flugzeug diskretisiert. Dazu wird ein Netz um die-

ses Gebiet erstellt. Die untenstehende Abbildung zeigt ein solches Netz im dreidimensionalen Fall.

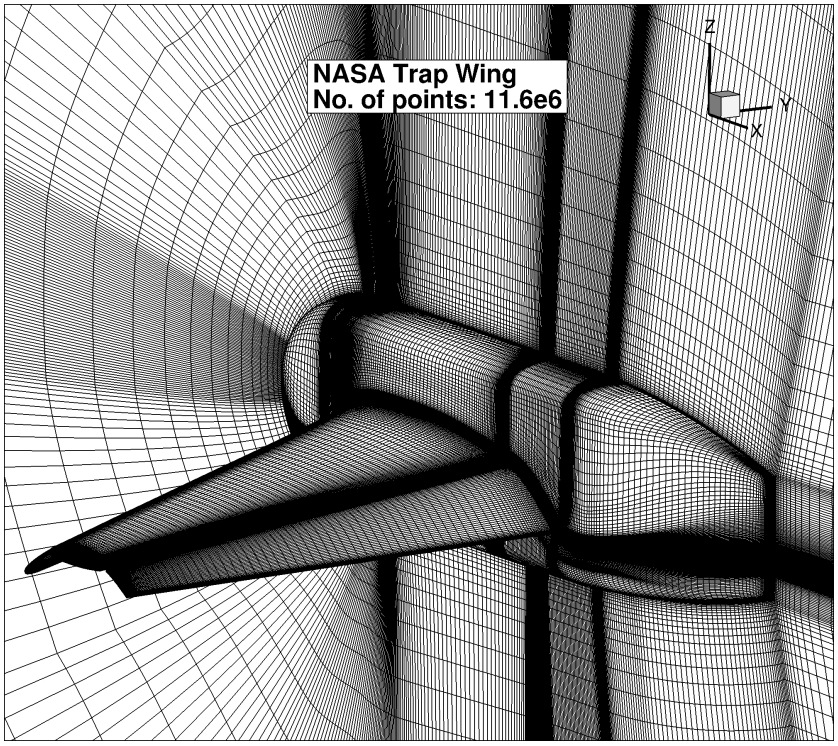


Abbildung 1.1: Netz um einen Teil eines Flugzeuges im dreidimensionalen Fall

Mithilfe dieses Netzes hat man somit eine endliche Anzahl von Punkten festgelegt, für die man eine Lösung approximieren möchte.
Dabei entsteht ein steifes, nichtlineares Problem mit einer großen Anzahl an Unbekannten. Das Mehrgitterverfahren ist dabei eine ein geeignetes Verfahren um dieses Problem zu lösen. Dabei liefert es eine gute Konvergenz und damit eine Lösung des Problems in vertretbarer Zeit.
Bei diesem Verfahren rechnet man nicht mehr nur auf einem Netz, sondern es wird eine Sequenz miteinander zusammenhängender, immer gröber werdender Netze erzeugt. Die Idee dieses Verfahren besteht darin, dass der Fehler auf dem feinen Netz bereits nach wenigen Iterationen glatt wird, d.h. die hochfrequenten Anteile des Fehlers werden stark gedämpft. Durch die Glättung kann der Fehler nun auch auf einem gröberen Gitter approximiert werden.

Zudem kann die Korrektur auf einem groben Netz durch die geringere Anzahl an Punkten schneller berechnet werden.
Hierzu benötigt man einerseits einen Projektionsoperator zur Restriktion vom feinen auf ein grobes Netz und einen Interpolationsoperator zur Prolongation der Ergebnisse vom groben auf die feinen Netze. Zudem muss ein passender Glätter gewählt bzw. konstruiert werden.
Die Netze zur Diskretisierung des Problems kann man dabei in 2 Gruppen aufteilen. Zum einen kann man strukturierte Netze erzeugen. Diese sind aufwändig in der Erzeugung, jedoch besitzt jedes Element einen Index, worüber man wiederum leicht die Nachbarn eines Elements bestimmen kann. Ihnen gegenüber stehen die unstrukturierten Netze, die zwar einfach zu erzeugen sind, aber es ist schwieriger aus ihnen ein gröberes Netz zu erzeugen. Vor allem bei komplexen dreidimensionalen Strukturen ist es deutlich schwieriger ein strukturiertes Netz zu erzeugen.
Ein weiterer Unterschied ist, dass in manchen Beispielen ein signifikanter Leistungsverlust eines Mehrgitterverfahrens von einem strukturierten zu einem unstrukturiertem Gitter erkennbar ist.
Im Folgenden ist der Vergleich für den Testfall auf dem $k\omega$-Modell (einer Art der Turbulenzmodellierung) mit einer Reynolds-Zahl von $6,210^6$, einer Mach-Zahl von $0,75$ und einem Anstellwinkel von $2,81°$.

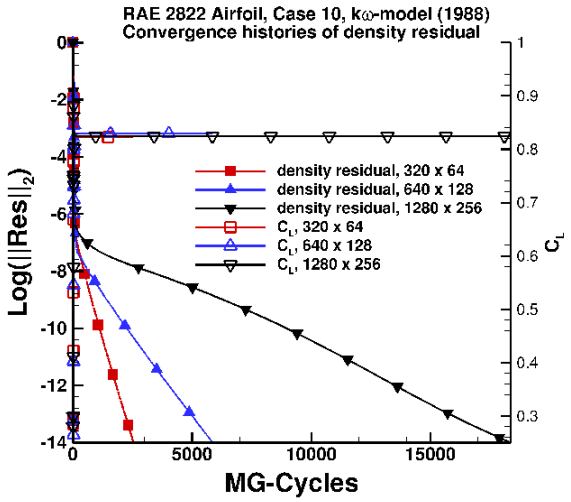


Abbildung 1.2: Fall 10 für das $k\omega$-Modell auf einem unstrukturierten Netz

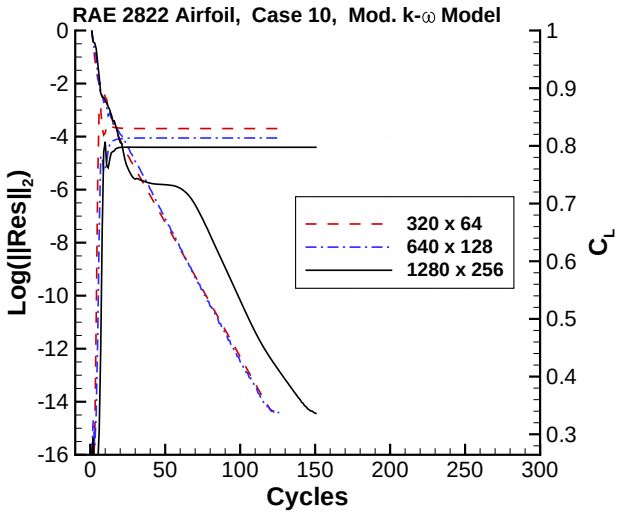


Abbildung 1.3: Fall 10 für das $k\omega$-Modell auf einem strukturierten Netz

Man sieht, dass das Verfahren auf dem strukturierten Netz bei allen gezeigten Netzgrößen deutlich schneller konvergiert. Im Rahmen dieser Arbeit soll auf Grundlage eines an den TAU-Code angelehnten Forschungscodes die Frage untersucht werden, ob man die Leistungsfähigkeit eines strukturierten Mehrgitters auf ein unstrukturiertes Mehrgitter übertragen kann. Dazu soll für ein Gitter, welches zwar formal unstrukturiert ist, aber nur aus Vierecken bzw. Hexaedern besteht, eine Sequenz von Grobgittern erzeugt werden.
Diese Grobgitter sollen denen in einem strukturierten Mehrgitterverfahren entsprechen. Im nächsten Schritt werden Interpolations- und Projektionsoperatoren, welche in strukturierten Verfahren Verwendung finden, implementiert. Die Ergebnisse dieser beiden Schritte sollen dann mit den Ergebnissen der unstrukturierten, agglomerationsbasierten Mehrgitterverfahren verglichen werden.
Der Aufbau dieser Arbeit ist dabei wie folgt. In Kapitel 2 sollen zunächst die theoretischen Grundlagen erklärt werden. Dazu werden in 2.1 und 2.2 das grundlegende mathematische Modell, die Navier-Stokes-Gleichungen und die Euler-Gleichungen, vorgestellt. Danach wird in Abschnitt 2.3 auf die Existenz und Eindeutigkeit einer Lösung der Navier-Stokes-Gleichungen eingegangen und der Begriff einer schwachen Lösung vorgestellt. In Abschnitt 2.4 wird die

Diskretisierung der in den ersten zwei Abschnitten des Kapitels vorgestellten partiellen Differentialgleichungen im Sinne des Finiten-Volumen-Verfahrens beschrieben. In Kapitel 3 folgt die Darstellung eines Mehrgitterverfahren. Hierzu soll zunächst die Grundidee des Mehrgitterverfahrens in Abschnitt 3.1 erläutert werden. Der folgende Abschnitt 3.2 beschäftigt sich mit den für das Mehrgitterverfahren essentiellen Gittern, auch Netze genannt. In Abschnitt 3.3 wird das verallgemeinerte Mehrgitterverfahren beschrieben. Die beiden folgenden Abschnitte behandeln die Komponenten des Mehrgitterverfahrens. Abschnitt 3.5 behandelt ein Verfahren zum Glätten des Residuums und 3.4 die Operatoren zum Übertragen des Residuums von einem Gitter auf ein nächstgröberes oder auf ein nächstfeineres. Das Kapitel 4 beschäftigt sich mit der Implementierung der neuen Ansätze in den bereits bestehenden Code. Hierbei wird in Abschnitt 4.1 das Verfahren zur Agglomeration der Grobnetze beschrieben. In Abschnitt 4.2 wird die Umsetzung der Operatoren beschrieben. Zuletzt erfolgt in Kapitel 5 der Vergleich der Ergebnisse dieser Arbeit mit dem bestehenden Verfahren und in Kapitel 6 der Ausblick auf weitere mögliche Ansatzpunkte zur Verbesserung des Verfahrens.

Kapitel 2

Mathematische Formulierung des Problems

Dieses Kapitel erläutert ein Modell zur Simulation der Strömung von Gasen und Flüssigkeiten. Dazu werden die zugrundeliegenden partiellen Differentialgleichungen vorgestellt und beschrieben. Eine Lösung dieser PDGL soll dabei nicht im klassischen Sinne gesucht werden, sondern es ist eine schwache Lösung gesucht.
Eine genaue Herleitung der Navier-Stokes-Gleichungen findet sich beispielsweise in [38]. Die folgenden Gleichungen und Definitionen beruhen größtenteils auf [3] Kapitel 2.

2.1 Navier-Stokes-Gleichungen

Der Bereich der CFD (computational fluid dynamics) beschäftigt sich mit der Modellierung der Strömung von Fluiden, also von Gasen und Flüssigkeiten. Hierbei werden einige Vereinfachungen angenommen, um die physikalischen Vorgänge in einem mathematischen Modell zu simulieren. Eine Annahme ist dabei, dass die Fluide, welche physikalisch betrachtet aus einer großen Anzahl von Molekülen oder Atomen bestehen, aufgrund ihrer hohen Dichte als Kontinuum approximiert werden können. Durch diese Annahme kann man für jeden Punkt physikalische Größen wie Dichte, Druck usw. bestimmen.

Eine weitere Annahme ist die Vernachlässigung der chemischen Reaktionen innerhalb eines Fluids. Eine Übersicht über diese Modellvereinfachungen findet sich z.B. in [14] S.3.
Als Betrachtungsweise wird die Eulersche Betrachtungsweise verwendet. Hierbei betrachtet man im Gegensatz zu Lagrangeschen Betrachtungsweise nicht die einzelnen Teilchen des Fluids, wie sie sich im Laufe der Zeit im Raum bewegen, sondern man wählt einen ortsfesten Raum und betrachtet auf diesem die Veränderungen. Dieses Teilgebiet des gesamten zu betrachteten Raumes nennt man auch Kontrollvolumen.

Definition 2.1.1. Kontrollvolumen
Sei ein Strömungsfeld auf einer Menge D mit $D \neq \emptyset$, $D \subset \mathbb{R}^m$ gegeben. Dann heißt eine beliebige, im Raum feste und abgeschlossene Menge Ω mit $\Omega \neq \emptyset$, $\Omega \subset D$ mit Rand $\partial\Omega$ Kontrollvolumen. Für ein Oberflächenelement dS sei zudem der Einheitsnormalenvektor $\overrightarrow{n}$ gegeben.

Dann ist das Flächenelement dS gegeben durch

$$dS = \sqrt{g(t)}d^k t f r x = \psi(t)$$

mit $g(t)$ als Gramscher Determinante. [12, S.167] Das Volumenelement dΩ steht für d$d^n x$ wobei n für die Dimension von Ω steht [12, S.189] Der Rand $\partial\Omega$ lässt sich wie folgt definieren. Sei X ein metrischer Raum und Ω eine Teilmenge von X. zudem sei $x \in X$. x heißt dann Randpunkt von Ω , wenn für jede ε-Umgebung von x ein Punkt in Ω und in $X \backslash \Omega$ befindet. Alle Randpunkte x bilden den Rand $\partial\Omega$ [11, S.9]
Die Gleichungen zur Beschreibung der Bewegungen von Gasen und Flüssigkeiten müssen drei fundamentale physikalische Prinzipien erfüllen. Diese sind das Massenerhaltungsgesetz, das Energieerhaltungsgesetz und das Impulserhaltungsgesetz.
Mithilfe des Massenerhaltungsgesetz wird die Kontinuitätsgleichung hergeleitet. Dazu betrachtet man Quader $S \subset \mathbb{R}^3$ mit Kantenlängen dx, dy, dz. Zudem seien die Dichte ρ, die Zeit t und die Geschwindigkeit $\overrightarrow{v}$ mit den Komponenten u, v, w gegeben, wobei diese die Geschwindigkeit respektive

in x-, y- und z-Richtung beschreiben. Für das Volumen des Quaders gilt $V = dx\,dy\,dz$.
Der Massenstrom q durch eine Oberfläche A beschreibt die Masse, welche durch diese Fläche innerhalb einer Zeiteinheit fließt. Er ist gegeben durch

$$q = \rho\,\overrightarrow{v}\,A$$

Der Massenfluss beschreibt den Massenstrom je Einheitsfläche und ist daher durch $\rho\,\overrightarrow{v}$ gegeben [32]. Der Massenfluss in x-Richtung ist somit durch ρu gegeben. Sei q_x^E der Eintrittsmassenstrom in x-Richtung. Somit ergibt sich

$$q_x^E = \rho\,u\,dy\,dz$$

Der Austrittsmassenfluss $\rho(x+dx)\,u$, welcher in x-Richtung nach dem Passieren der Quaders auftritt, kann mithilfe der Taylor-Reihe approximiert werden und lautet daher

$$\rho(x+dx)\,u = \rho(x)\,u + (\frac{\partial(\rho u)}{\partial x}\,dx)$$

Somit ergibt sich als Austrittsmassenstrom q_x^A

$$q_x^A = \rho\,u\,dy\,dz + (\frac{\partial(\rho u)}{\partial x}\,dx)dy\,dz$$

Somit erhält man als Massenflussbilanz in x-Richtung, welche sich als Differenz von q_x^E und q_x^A ergibt

$$q_x^E - q_x^A = \rho\,u\,dy\,dz - (\rho\,u\,dy\,dz + (\frac{\partial(\rho u)}{\partial x}\,dx)dy\,dz) = -(\frac{\partial(\rho u)}{\partial x}\,dx)dy\,dz$$

Diese Überlegung kann man analog für die y- und z-Richtung durchführen und erhält als Massenflussbilanz in y-Richtung

$$-(\frac{\partial(\rho v)}{\partial y}\,dy)dx\,dz$$

bzw. in z-Richtung

$$-(\frac{\partial(\rho w)}{\partial z}\,dz)dx\,dy$$

Als Massenflussbilanz in alle Richtungen ergibt sich dann

$$-(\frac{\partial(\rho u)}{\partial x}\,dx)dy\,dz - (\frac{\partial(\rho v)}{\partial y}\,dy)dx\,dz - (\frac{\partial(\rho w)}{\partial z}\,dz)dx\,dy$$

, was sich zu

$$-\mathrm{div}(\rho\overrightarrow{v})dx\,dy\,dz$$

zusammenfassen lässt. Die Divergenz ist dabei für ein beliebiges Vektorfeldes

$$A = \begin{pmatrix} a_1(x,y,z) \\ a_2(x,y,z) \\ a_3(x,y,z) \end{pmatrix}$$

durch

$$divA = \frac{\partial a_1}{\partial x} + \frac{\partial a_2}{\partial y} + \frac{\partial a_3}{\partial z}$$

gegeben. Nach dem Massenerhaltungsgesetz muss die Massenflussbilanz der Veränderung der Masse innerhalb des Quaders entsprechen. Diese Änderung wird durch die Veränderung der Dichte im Laufe der Zeit verursacht und kann somit durch

$$[\frac{\rho(t+\Delta t) - \rho(t)}{\Delta t}] \cdot V \approx \frac{\partial \rho}{\partial t} dx\,dy\,dz$$

ausgedrückt werden. Somit erhält man

$$\frac{\partial \rho}{\partial t} dx\,dy\,dz = -\mathrm{div}(\rho\overrightarrow{v})dx\,dy\,dz$$

oder auch

$$\frac{\partial \rho}{\partial t} + \mathrm{div}(\rho\overrightarrow{v}) = 0 \tag{2.1}$$

Die Gleichung 2.1 drückt somit aus, dass innerhalb eines Volumens die Summe der ein- und ausfließenden Massenströme gleich der zeitlichen Veränderung der Masse (welche durch die Veränderung der Dichte entsteht) entspricht. Diese Gleichung wird auch Kontinuitätsgleichung genannt.
Integriert man diese Gleichung über ein beliebiges Kontrollvolumen Ω hat 2.1 folgende Form.

$$\int_\Omega \frac{\partial \rho}{\partial t} \, \mathrm{d}\Omega + \int_\Omega \operatorname{div}(\rho \overrightarrow{v}) \, \mathrm{d}\Omega = 0 \tag{2.2}$$

Im ersten Term kann man die Integration und Differentiation vertauschen, da $\mathrm{d}\Omega$ unabhängig von t ist. Mithilfe des Gaußschen Integralsatzes lässt sich der zweite Term in ein Oberflächenintegral verwandeln. Somit erhält man

$$\frac{\partial}{\partial t} \int_\Omega \rho \, \mathrm{d}\Omega + \oint_{\partial\Omega} \rho \, \overrightarrow{v} \cdot \overrightarrow{n} \, \mathrm{d}S = 0 \tag{2.3}$$

Dabei bezeichnet $\cdot$ das Skalarprodukt. Für zwei Vektoren $\overrightarrow{a}$, $\overrightarrow{b} \in \mathbb{R}^n$ ergibt sich

$$\overrightarrow{a} \cdot \overrightarrow{b} = \sum_{i=1}^{n} a_i b_i$$

[13, S.188]

Nun betrachtet man den Impulserhaltungssatz zum Herleiten der Impulsgleichung. Dieser sagt aus, dass die zeitliche Änderung des Impulses gleich der Summe der einwirkenden Kräfte ist. Im Gegensatz dazu sagt der oben verwendete Massenerhaltungssatz aus, dass die Änderung der Masse Null ist. Aus diesem Grund kann man zwar einen Teil der Impulsgleichungen analog zur Kontinuitätsgleichung herleiten, jedoch muss man noch die wirkenden Kräfte, hier zunächst mit $\overrightarrow{F}$ bezeichnet, beachten. Verwendet man nun $\rho \cdot \overrightarrow{v}$ statt ρ, so erhält man ähnlich wie in Gleichung 2.3

$$\frac{\partial}{\partial t} \int_\Omega \rho \, \overrightarrow{v} \, \mathrm{d}\Omega + \oint_{\partial\Omega} \rho \, \overrightarrow{v} \cdot \overrightarrow{v} \cdot \overrightarrow{n} \, \mathrm{d}S = \overrightarrow{F}$$

Die Kräfte, welche auf das Volumen wirken können dabei einer der beiden folgenden Arten angehören. Zum einen gibt es externe Kräfte, welche die Masse des Volumens beeinflussen. Dies können z.B. die Schwerkraft oder die Corioliskraft sein. In diesem Modell können sie jedoch vernachlässigt werden. Zum anderen gibt es Kräfte, welche durch die Interaktion der Oberfläche des Volumens mit anderen Fluidteilchen entstehen. Dies sind zum einen der Druck p, welcher auf das Volumen einwirkt und die Scher- und Normalspannung, welche durch die Reibung erzeugt werden. In Integralform haben sie

die Form

$$-\oint_{\partial} p\,\overrightarrow{n}\,\mathrm{d}S \tag{2.4}$$

und

$$\oint_{\partial} \sigma \cdot \overrightarrow{n}\,\mathrm{d}S \tag{2.5}$$

mit dem Cauchyschen Spannungstensor σ

$$\sigma = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix}$$

Man bezeichnet $\sigma_x, \sigma_y, \sigma_z$ als Normalspannungen und $\tau_{xy}, \tau_{xz}, \tau_{yz}, \tau_{zx}, \tau_{zy}, \tau_{yx}$ als Scherspannungen. τ_{ij} bedeutet dabei, dass die Spannung auf einer Ebene senkrecht zur i-Achse in Richtung j-Achse wirkt. Die Normalspannungen σ_i haben die Bedeutung eines τ_{ii}.
Um die Werte des Spannungstensors zu berechnen,trifft man die Annahme, dass es sich bei dem zu untersuchenden Fluid um eine newtonsches Fluid handelt. Als newtonsches Fluid wird dabei eine Flüssigkeit oder ein Gas bezeichnet, für welches man annimmt, dass es ein lineare Viskosität ausweist. Beispiele für newtonsche Fluide sind Wasser oder Luft und für nicht-newtonsche Fluide Blut oder Treibsand. Unter der Annahme eines newtonschen Fluids gilt dann die Gleichung

$$\tau = \eta \frac{dv}{dy}$$

wobei η als dynamische Viskosität bezeichnet wird und abhängig vom Medium einen spezifischen Wert annimmt.
Zudem ist der Spannungstensor durch Anwendung des Drehimpulssatzes symmetrisch, es gilt also $\tau_{ij} = \tau_{ji}$. Somit erhält man

$$\begin{aligned} \sigma_x &= \lambda \mathrm{div}\,\overrightarrow{v} + 2\eta \frac{\partial u}{\partial x} \\ \sigma_y &= \lambda \mathrm{div}\,\overrightarrow{v} + 2\eta \frac{\partial v}{\partial y} \\ \sigma_z &= \lambda \mathrm{div}\,\overrightarrow{v} + 2\eta \frac{\partial w}{\partial z} \end{aligned}$$

$$\begin{aligned}\tau_{xy} &= \eta(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x}) \\ \tau_{xz} &= \eta(\frac{\partial u}{\partial z} + \frac{\partial w}{\partial x}) \\ \tau_{yz} &= \eta(\frac{\partial v}{\partial z} + \frac{\partial w}{\partial y})\end{aligned} \tag{2.6}$$

Der Parameter λ ist die erste Lamé-Konstante. Außerdem nimmt man die Hypothese von Stokes an die zum Verhältnis

$$\lambda + \frac{2}{3}\eta = 0$$

führt. Dadurch kann man die Lamé-Konstante mit der Viskosität in den Termen für σ_x, σ_y und σ_z ersetzen und erhält

$$\begin{aligned}\sigma_x &= -\frac{2}{3}\eta \mathrm{div}\,\overrightarrow{v} + 2\eta\frac{\partial u}{\partial x} \\ \sigma_y &= -\frac{2}{3}\eta \mathrm{div}\,\overrightarrow{v} + 2\eta\frac{\partial v}{\partial y} \\ \sigma_z &= -\frac{2}{3}\eta \mathrm{div}\,\overrightarrow{v} + 2\eta\frac{\partial w}{\partial z}\end{aligned} \tag{2.7}$$

Zuletzt betrachtet man das Energieerhaltungsgesetz. Dazu wird der erste Hauptsatz der Thermodynamik verwendet, der aussagt, dass eine Veränderung der Energie des Volumens, nur durch Veränderungen der Wärmeenergie oder durch Veränderung der Arbeit erzeugt werden kann. Die Gesamtenergie des Volumens besteht aus der Summe der Energie im Ruhezustand, der inneren Energie, und der Bewegungsenergie $\frac{m||\overrightarrow{v}||_2^2}{2}$. Mithilfe der spezifischen Größen, das heißt in Bezug auf die Masse, erhält man mit der spezifischen Gesamtenergie E und der spezifischen inneren Energie e die folgende Gleichung.

$$E = e + \frac{||\overrightarrow{v}||_2^2}{2}$$

Die zeitliche Änderung von E, lässt sich wie bei den vorherigen Erhaltungssätzen durch

$$\frac{\partial}{\partial t}\int_\Omega \rho\, E \,\mathrm{d}\Omega + \oint_{\partial\Omega} \rho\, E \cdot \overrightarrow{v} \cdot \overrightarrow{n}\,\mathrm{d}S = F$$

ausdrücken. Die Variable F steht in diesen Fall für die Änderung der Wärme

und Arbeit. Die Arbeit entsteht durch die Kräfte, die schon beim Impulserhaltungssatz benannt wurden, also der Druck und die Scher- und Normalspannung, jedoch muss hier auch noch die Geschwindigkeit betrachtet werden. Dementsprechend sind die Terme

$$- \oint_{\partial} p \overrightarrow{n} \overrightarrow{v} \, \mathrm{d}S \tag{2.8}$$

und

$$\oint_{\partial} \sigma \cdot \overrightarrow{n} \overrightarrow{v} \, \mathrm{d}S \tag{2.9}$$

in F enthalten. Hinzu kommt aber noch die Wärmestromdichte $\overrightarrow{q}$, welche mithilfe des Fourierschen Gesetzes beschrieben werden kann.

$$\overrightarrow{q} = -k \mathrm{grad} T$$

$\overrightarrow{q}$ steht für die Wärmestromdichte, k ist der Wärmeleitungskoeffizient und der Gradient der Temperatur T in Kelvin grad T ist der Vektor

$$\text{grad } T = \nabla T = \begin{pmatrix} \frac{\partial T}{\partial x} \\ \frac{\partial T}{\partial y} \\ \frac{\partial T}{\partial z} \end{pmatrix}$$

Als Integral enthält man dadurch

$$\oint_{\partial\Omega} k(\text{grad } T \cdot \overrightarrow{n}) \, \mathrm{d}S$$

Durch das Ausnutzen der Beziehung zwischen der spezifischen Enthalpie h und e

$$h = e + \frac{p}{\rho} \tag{2.10}$$

bzw. zwischen der spezifischen Totalenthalpie H und E

$$H = h + \frac{||\overrightarrow{v}||_2^2}{2} = e + \frac{p}{\rho} + \frac{||\overrightarrow{v}||_2^2}{2} = E + \frac{p}{\rho} \tag{2.11}$$

erhält man

$$\oint_{\partial\Omega} \rho \, E \cdot \overrightarrow{v} \cdot \overrightarrow{n} \, \mathrm{d}S = \oint_{\partial\Omega} \rho \, H \cdot \overrightarrow{v} \cdot \overrightarrow{n} \, \mathrm{d}S - \oint_{\partial\Omega} p \cdot \overrightarrow{v} \cdot \overrightarrow{n} \, \mathrm{d}S$$

Dadurch kürzt sich der Term 2.8 auf beiden Seiten weg. Zusammengefasst führt dies dann zu folgendem Gleichungssystem.

Definition 2.1.2. Navier-Stokes-Gleichungen
Die folgende Gleichung

$$\frac{\partial}{\partial t}\int_{\Omega}\overrightarrow{W}\,\mathrm{d}\Omega + \oint_{\partial\Omega}(\overrightarrow{F}_C - \overrightarrow{F}_V)\,\mathrm{d}S = \overrightarrow{0} \tag{2.12}$$

mit den folgenden Variablen im dreidimensionalen Fall

$$\overrightarrow{W} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{bmatrix}$$

$$\overrightarrow{F}_C = \begin{bmatrix} \rho(\overrightarrow{v}\cdot\overrightarrow{n}) \\ \rho u(\overrightarrow{v}\cdot\overrightarrow{n}) + n_x p \\ \rho v(\overrightarrow{v}\cdot\overrightarrow{n}) + n_y p \\ \rho w(\overrightarrow{v}\cdot\overrightarrow{n}) + n_z p \\ \rho H(\overrightarrow{v}\cdot\overrightarrow{n}) \end{bmatrix}$$

mit $(\overrightarrow{v}\cdot\overrightarrow{n}) = n_x u + n_y v + n_z w$ und H wie in 2.11,

$$\overrightarrow{F}_V = \begin{bmatrix} 0 \\ n_x\sigma_x + n_y\tau_{xy} + n_z\tau_{xz} \\ n_x\tau_{yx} + n_y\sigma_y + n_z\tau_{yz} \\ n_x\tau_{zx} + n_y\tau_{zy} + n_z\sigma_z \\ \sum_{i\in\{x,y,z\}} n_i(u\tau_{ix} + v\tau_{iy} + w\tau_{iz} + k\frac{\partial T}{\partial i}) \end{bmatrix}$$

wobei τ_{ii} als σ_i zu interpretieren ist und die $\sigma_i und \tau_{ij}$ wie in 2.7 und 2.6 definiert sind. Dieses Gleichungssystem heißt für ein newtonsches Fluid Navier-Stokes-Gleichungen.

Der Vektor $\overrightarrow{F}_C$ beschreibt dabei den konvektiven Teil des Flusses und $\overrightarrow{F}_V$ den viskosen Teil des Flusses.

Das obenstehende Gleichungssystem bestehen aus einem System von 5 Gleichungen. Da es aber insgesamt 7 Unbekannte gibt, müssen je nach modelliertem Fluid noch 2 weitere Gleichungen ergänzt werden.
In der Modellierung der Strömung von Luft um ein Flugzeug wird ein perfektes Gas angenommen. Das bedeutet, dass man einerseits annimmt, dass die thermische Zustandsgleichung

$$p = \rho RT \tag{2.13}$$

mit R als spezifischer Gaskonstante gilt. Zudem gilt für ein perfektes Gas, dass der spezifische Wärmekoeffizient für gleichbleibenden Druck c_p und für ein gleichbleibendes Volumen c_v konstant ist. Diese fasst man in der Kennzahl

$$\gamma = \frac{c_p}{c_v}$$

zusammen, die für Luft der Wert $1,4$ besitzt. Zudem gelten

$$e = c_v T \qquad R = c_p - c_v$$

Damit kann man 2.13 wie folgt umformen

$$\begin{aligned} p = \rho RT = \rho(c_p - c_v)T = \rho\frac{c_v(c_p - c_v)}{c_v}T = \rho(\gamma - 1)c_v T \\ = (\gamma - 1)\rho\, e = (\gamma - 1)\rho(E - \frac{\|\overrightarrow{v}\|_2^2}{2}) \end{aligned}$$

Zudem wird der Viskositätskoeffizient η, welcher für den Spannungstensor benötigt wird, für ein perfektes Gas mithilfe der Sutherland-Formel definiert.

$$\eta = \frac{1,45T^{\frac{3}{2}}}{T + 110} \cdot 10^{-6} \tag{2.14}$$

Statt der dimensionshafteten Größen in 2.12 kann man die Gleichung umstellen, in dem man die Größen durch dimensionslose Kenngrößen ersetzt. Dazu verwendet man das Buckinghamsche Π-Theorem. Dies sagt aus, dass man ein physikalisches System dimensionsbehafteten Bezugsgrößen in ein System mit einer bestimmten Anzahl unabhängiger, dimensionsloser Kenngrößen überführen kann. Dabei ist die Anzahl der benötigten Kenngrößen

a_K durch $a_K = a_E - a_D$ bestimmt, wobei a_E die Anzahl der Einflussgrößen und a_D der Rang der Dimensionsmatrix D ist, welche die unabhängigen Einflussgrößen in Zusammenhang mit ihren Bezusgrößen darstellt. Im Fall der Navier-Stokes-Gleichungen hat man ein System mit den vier Bezugsgrößen, nämlich Bezugslänge, Bezugsgeschwindigkeit, Bezugsdichte und Bezugstemperatur [14], S.55, Somit erhält man für die Einflussgrößen Dichte, Druck, Temperatur, Geschwindigkeit und spezifische Gaskonstante die folgende Dimensionsmatrix.

	ρ	p	T	$\overrightarrow{v}$	R
m	0	0	0	0	0
$\frac{m}{s}$	0	2	0	1	2
$\frac{kg}{m^3}$	1	1	0	0	0
K	0	0	1	0	-1

Diese Matrix hat den Rang 3, somit benötigt man 2 dimensionslose Kenngrößen. Dafür betrachtet man die Basen des Nullraums

$$\begin{pmatrix} 1 \\ -1 \\ 0 \\ 2 \\ 0 \end{pmatrix}, \qquad \begin{pmatrix} 1 \\ -1 \\ 1 \\ 0 \\ 1 \end{pmatrix} \tag{2.15}$$

, wodurch man zu den Beziehungen

$$1 = \frac{\rho(\overrightarrow{v})^2}{p}, \quad 1 = \frac{\rho T R}{p}$$

kommt.
Sind nun der Referenz-Druck p_{Ref}, die Referenzdichte ρ_{Ref} und die Referenztemperatur T_{Ref} gegeben, so können die Referenzgeschwindigkeit $\overrightarrow{v}_{Ref}$ und die spezifische Referenzgaskonstante R_{Ref} durch

$$\overrightarrow{v}_{Ref} = \sqrt{\frac{p_{Ref}}{\rho_{Ref}}}, \quad R_{Ref} = \frac{p_{Ref}}{\rho_{Ref} T_{Ref}}$$

ausgedrückt werden. Mit diesen Referenzgrößen kann man zudem die wei-

teren Variablen in nichtdimensionale Größen überführen. Somit erhält man eine zu 2.12 ähnliche Gleichung, in der lediglich die Variablen mit ihren dimensionslosen äquivalenten Größen ersetzt wurden und der viskose Teil den Vorfaktor $\frac{1}{Re}$ erhält (siehe [25, S.37ff.]). Re steht für die Reynolds-Zahl, die als

$$Re = \frac{\rho \overrightarrow{v} L}{\eta}$$

mit L als charakteristischer Länge definiert ist. Bei einem Flügel kann dies beispielsweise die Länge des Profils sein.
Eine andere häufige Wahl ist Prandtl-Zahl Pr mit

$$Pr = \frac{\eta c_p}{k}$$

Der Wärmeleitungskoeffizient k kann so mithilfe von

$$k = c_p \frac{\eta}{Pr}$$

berechnet. c_p kann dabei mithilfe von R und γ berechnet werden

$$c_p = \frac{\gamma R}{\gamma - 1}$$

und hat somit für Luft den Wert $3,5R$.
Die Prandtl Zahl beträgt im Fall von Luft $0,72$, die Reynolds Zahl Re ergibt sich durch die physikalischen Bedingungen.
Mit 2.12 können nun theoretisch sowohl turbulente als auch laminare Strömungen berechnet werden. Jedoch tritt bei turbulenten Strömungen mit hohen Reynolds-Zahlen das Problem auf, dass selbst die Rechenleistung moderner Computer nicht ausreicht, um sie direkt zu berechnen (auch DNS, direkte numerische Simulation genannt). Ein Ansatz besteht darin, dass die Navier-Stokes-Gleichungen in eine Grundströmung und Schwankungen eingeteilt werden. Dazu teilt man Druck p und die Geschwindigkeitskomponenten u, v, w

$$u = \bar{u} + u' \qquad v = \bar{v} + v' \qquad w = \bar{w} + w' \qquad p = \bar{p} + p'$$

wobei der Überstrich die gemittelte Größe und Hochstrich die turbulen-

te Größe kennzeichnet. Um die Grundströmung zu berechnen mittelt man die Navier-Stokes-Gleichungen bezüglich der Zeit. Dies führt zu den sogannten Reynoldsgemittelte Navier-Stokes Gleichungen (auch als RANS „reynolds averaged Navier Stokes“ bezeichnet). Eine Beschreibung findet sich im folgenden Abschnitt 2.1.1.
Die Schwankungen werden durch sogenannte Turbulenzmodelle beschrieben. Ein solches mögliches Modell wird kurz im Abschnitt 2.1.2 beschrieben.

2.1.1 Reynoldsgemittelte Navier-Stokes-Gleichungen

Je nach Art der Turbulenzen gibt es verschiedene Arten die Gleichungen zu mitteln. Für stationäre Turbulenzen verwendet man die zeitliche Mittelung

$$\bar{s} = \lim_{T\to\infty} \frac{1}{T} \int_t^{t+T} s \, \mathrm{d}t$$

mit $s \in \{u, v, w\}$
Diese Mittelung führt für die Navier-Stokes-Gleichungen zu eben diesen und folgendem zusätzlichem Term.

$$\tau_{i,j}^R = -\rho \bar{v_i'} \bar{v_j'} = \rho(\bar{v_i}\bar{v_j} - \bar{v_i}\bar{v_j})$$

$\tau_{i,j}^R$ wird dabei auch Reynolds-Spannungstensor genannt.
Im dreidimensionalen Fall hat der Tensor folgende Form

$$\rho \bar{v_i'} \bar{v_j'} = \begin{bmatrix} \rho\overline{(v_1')^2} & \rho\overline{v_1'v_2'} & \rho\overline{v_1'v_3'} \\ \rho\overline{v_2'v_1'} & \rho\overline{(v_2')^2} & \rho\overline{v_2'v_3'} \\ \rho\overline{v_3'v_1'} & \rho\overline{v_3'v_2'} & \rho\overline{(v_3')^2} \end{bmatrix}$$

Diese Gleichung wird auch Reynolds-Spannungstensor genannt.
Die laminare viskose Spannung wird durch folgende Gleichung beschrieben.

$$\overline{\tau_{ij}} = 2\mu\overline{S}_{ij} = \mu(\frac{\partial\overline{v}_i}{\partial x_j} + \frac{\partial\overline{v}_j}{\partial x_i})$$

Um das Gleichungssystem, welches durch diese zusätzliche Gleichung mehr Unbekannte hat, lösen zu können, müssen weitere Gleichungen ergänzt wer-

den. Dabei gibt es verschiedene Ansätze, die sogenannten Turbulenzmodelle, welche im folgenden Abschnitt kurz erklärt werden.

2.1.2 Turbulenzmodelle

Turbulenzmodelle dienen dazu die neuen Unbekannten aus der im vorherigen Abschnitt vorgestellten Reynolds-Spannung zu approximieren. Zwei der bekanntesten Modelle sind dabei das Modell von Spalart und Allmaras und das $k\omega$-Modell.
Beide Modelle beruhen auf der Hypothese von Boussinesq. Hierbei wird angenommen, dass sich die turbulente Scherspannung proportional zur durchschnittlichen Rate der Dehnung verhält. Für die RANS-Gleichung hat sie die folgende Form.

$$\tau_{i,j}^{R} = -\rho v_i \bar{v}_j = 2\mu_T \bar{S}_{i,j} - \frac{2}{3}\rho K \delta_{i,j}$$

Der Faktor μ_T wird (turbulente) Wirbelviskosität genannt. Durch die Anwendung der Boussinesq-Annahme auf die Navier-Stokes-Gleichungen verändert sich zudem der Viskositätskoeffizient η zu

$$\eta = \eta_L + \eta_T$$

Der laminare Viskositätskoeffizient η_L wird mit der Sutherland Formel 2.14 berechnet. Außerdem ändert sich auch der Wärmeleitungskoeffizient k, da für ihn im turbulenten Fall die Gleichung

$$k = k_L + k_T$$

angenommen wird. Für den turbulenten Teil k_T gilt

$$k_T = c_P \frac{\mu_T}{Pr_T}$$

Der Parameter c_P ist der Wärmekoeffizient, welcher bei einem gleichbleibenden Druck gilt und Pr_T ist die turbulente Prandtl-Zahl, welche für Luft bei $0,9$ liegt.

Analog gilt für den laminaren Teil

$$k_L = c_P \frac{\mu_L}{Pr_L}$$

Dabei nimmt Pr_L für das Medium Luft den Wert $0,72$ an.
$\overline{S}_{i,j}$ steht für die gemittelten Komponenten des Tensors der Dehnungsgeschwindigkeit und wird wie im vorherigen Abschnitt beschrieben berechnet. Somit müssen nur noch die Wirbelviskosität μ_T und die turbulente kinetische Energie K bestimmt werden. Dies geschieht mithilfe eines Turbulenzmodells. Diese Modelle können u.a. aus algebraischen Beziehungen, einer oder mehreren Gleichungen bestehen.
Das negative Spalart Allmaras-Modell ist ein Vertreter der Eingleichungsmodelle und ist in [1] beschrieben. Da sich die Arbeit aber nur mit laminaren Strömungen beschäftigt, sei hier lediglich der Verweis gegeben.

2.1.3 Randbedingungen

Abschließend sollen noch die Randbedingungen angegeben werden. Innerhalb der Simulation eines Flugzeugflügels kommen zwei Randbedingungen vor, zum einen eine Fernfeld-Bedingung („farfield“) am äußeren Rand und eine Wand am Rand zum Bauteil selbst.
Eine Fernfeld-Bedingung ist dabei dann gegeben, wenn man nur durch die Begrenzung des zu modellierenden Raumes auf eine Grenze stößt, es aber in Wirklichkeit noch viel weiter gehen würde. Dies modelliert man, in dem man folgende Annahme trifft.

$$\overrightarrow{W}_a = (\rho_\infty, \cos\alpha\rho_\infty u_\infty, 0, \sin\alpha\rho_\infty u_\infty, \rho_\infty E_\infty)^T \tag{2.16}$$

für einen gegebenen Anstellwinkel α und am Randpunkt a.
Für die feste Wand, welche das Bauteil darstellt, gilt folgende Bedingung

$$u = v = w = 0$$

Das heißt, dass hier die Geschwindigkeitskomponenten gleich Null sind.

2.2 Euler-Gleichungen

Eine Vereinfachung der Navier-Stokes-Gleichungen ist die Betrachtung einer reibungsfreien, elastischen Flüssigkeit. In diesem Fall wird der Vektor der viskosen Flüssen mit Null gleichgesetzt und man hält die folgenden Gleichungen.

Definition 2.2.1. Euler-Gleichungen Die folgende Gleichung

$$\frac{\partial}{\partial t}\int_{\Omega}\overrightarrow{W}\,\mathrm{d}\Omega+\oint_{\partial\Omega}\overrightarrow{F}_C\,\mathrm{d}S=\overrightarrow{0} \tag{2.17}$$

zusammen mit $\overrightarrow{W}$ und $\overrightarrow{F}_C$ wie in 2.12 definiert, heißen Euler-Gleichungen.

Auch die Anfangs- und Randbedingungen ändern sich. Für eine feste Wand gilt nun

$$V\equiv\overrightarrow{v}\cdot\overrightarrow{n}=0$$

2.3 Lösbarkeit der Navier-Stokes-Gleichungen

Die Existenz, Eindeutigkeit und Regularität der Navier-Stokes-Gleichungen und Euler-Gleichungen im dreidimensionalen Raum gehören bisher zu den ungelösten Problemen. Im Falle der Navier-Stokes-Gleichung sind sie sogar von so großer Bedeutung, dass sie zu den Millenium-Problemen gehören, für deren Lösung jeweils ein Preisgeld von 1 Million Dollar vom Clay-Institut ausgelobt wurde. Fefferman beschreibt in [10] das Problem und den bisherigen Stand der Forschung. Bei bestimmten, kleinen Startgeschwindigkeiten oder in kleinem Betrachtungszeitraum können bereits Aussagen zu Existenz und Regularität gemacht werden. Für das allgemeine Problem gilt dies aber bisher noch nicht. Aus diesem Grund betrachtet man eine schwache Lösung. Diese stellt eine Erweiterung der klassischen Lösung einer Differentialgleichung dar und soll kurz in diesem Abschnitt vorgestellt werden. Die Idee ist, dass man eine schwache Lösung konstruiert und nachweist, dass diese immer glatt ist. So wurde bereits in [28] gezeigt, dass immer mindestens eine schwache Lösung der Navier-Stokes-Gleichung existiert. Zur Eindeutigkeit konnte aber bisher noch keine Aussage gemacht werden.

Zur Betrachtung der schwachen Lösung soll zunächst das folgende Problem untersucht werden:
Gegeben sei $f \in C([a,b])$. Gesucht sei u, sodass gilt

$$\begin{cases} -u'' + u = f \text{ auf } [a,b], \\ u(a) = u(b) = 0. \end{cases} \tag{*}$$

Eine klassische Lösung (mit der Ableitung als punktweiser Limes des Differenzenquotienten) wäre eine Funktion $u \in C^2([a,b])$, welche die oben stehenden Bedingungen erfüllt. Möchte man aber einen weiter gefassten Lösungsbegriff, so kann die obenstehende Gleichung mit $\varphi \in C^1([a,b])$ multiplizieren und partiell integrieren. Somit erhält man

$$\int_a^b u' \, \varphi' \, dx + \int_a^b u \, \varphi \, \mathrm{d}x = \int_a^b f \, \varphi \, \mathrm{d}x \tag{**}$$

für alle $\varphi \in C^1([a,b])$ mit $\varphi(a) = \varphi(b) = 0$.
Für eine Funktion $u \in C^2([a,b])$, $u(a) = u(b) = 0$, die (**) erfüllt, folgt zudem mithilfe von partieller Integration

$$\int_a^b (-u'' + u - f) \, \varphi \, \mathrm{d}x = 0$$

für alle $\varphi \in C^1([a,b])$ mit $\varphi(a) = \varphi(b) = 0$ und somit auch für alle $\varphi \in C_c^1((a,b))$ Es gilt (siehe [6], S.110) $-u'' + u = f$ fast überall auf (a,b) und damit auch auf $[a,b]$, da $u \in C^2([a,b])$. Daher erfüllt u auch (*).

Man erkennt, dass man nun nur noch eine Funktion $u \in C^1([a,b])$ benötigt, die die Gleichung erfüllt. Dies motiviert den auf eine offene Teilmenge von $\mathbb{R}^n$ erweiterten Begriff der schwachen (partiellen) Ableitung.

Notation 2.3.1. Sei Ω eine offene Teilmenge von $\mathbb{R}^n$ und $u \in L^1_{loc}(\Omega)$. $L^1_{loc}(\Omega)$ bezeichnet dabei die Menge der Funktionen $v : \Omega \to \mathbb{R}$, die für jede kompakte Teilmenge $K \subset \Omega$ in $L^1(K)$ sind, wobei Funktionen, die sich nur auf einer Nullmenge unterscheiden, miteinander identifiziert werden. $\alpha \in \mathbb{N}_0^n$ sei ein Multiindex, d.h. $\alpha = \{\alpha_1, \alpha_2, \cdots, \alpha_n\}$ und $|\alpha| = \alpha_1 + \alpha_2 + \cdots + \alpha_n$.

Zudem gelte

$$D^\alpha u = \frac{\partial^{|\alpha|} u}{\partial x_1^{\alpha_1} \dots \partial x_n^{\alpha_n}} = \frac{\partial^{\alpha_1}}{\partial x_1^{\alpha_1}} \cdots \frac{\partial^{\alpha_n}}{\partial x_n^{\alpha_n}} u$$

Definition 2.3.1. schwache (partielle) Ableitung
Eine Funktion $v \in L^1_{loc}(\Omega)$ heißt schwache (partielle) Ableitung von u, wenn gilt

$$\int_\Omega u D^\alpha \varphi \, \mathrm{d}x = (-1)^{|\alpha|} \int_\Omega v \, \varphi \, \mathrm{d}x \text{ für alle } \varphi \in C_0^\infty(\Omega) \tag{2.18}$$

Gilt dies, so schreibt man $D^\alpha u = v$. Die $\varphi \in C_0^\infty(\Omega)$ nennt man auch Testfunktionen.

Diese Funktionen kann man im sogenannten Sobolev-Raum zusammenfassen.

Definition 2.3.2. Sobolev-Raum Sei Ω eine offene Teilmenge des $\mathbb{R}^n$. Der Sobolev-Raum $W^{k,p}(\Omega)$ besteht aus allen Funktionen $u \in L^p(\Omega)$, sodass für jeden Multi-Index α mit $|\alpha| \leq k$ die schwache Ableitung $D^\alpha u$ existiert und in $\in L^p(\Omega)$ liegt, d.h. $W^{k,p}(\Omega) = \{u \in L^p(\Omega) : D^\alpha u \in L^p(\Omega), |\alpha| \leq k\}$

Der Sobolevraum $W^{k,p}(\Omega)$ besteht also aus $u \in L^p(\Omega)$, die schwache partielle Ableitungen bis zur Ordnung k besitzen, welche zudem auch zu $L^p(\Omega)$ gehören.
Eine (partielle) Differentialgleichung kann also mittels Multiplikation mit einer beliebigen Testfunktion und anschließender partieller Integration in eine Form gebracht werden, so dass ein erweiterter Lösungsbegriff anwendbar ist. Eine Funktion, welche aus den Sobolevräumen stammt und diese neue Gleichung erfüllt, nennt man schwache Lösung.
Da die Sobolevräume aber aus L^p Funktionen bestehen, die nur eine Äquivalenzklasse von Funktionen darstellen und somit auf Nullmengen geändert werden können, könnten Randwerte beliebig geändert werden. Da man aber Sobolevräume für die Lösung von Randwertproblemen nutzen möchte, muss man den Begriff der Randwerte auf Sobolovräume erweitern. Dazu dient das folgende Theorem

Theorem 2.3.1. $\Omega \subset \mathbb{R}^n$ sei eine offene Menge mit kompaktem Rand, wobei $\bar{\Omega} \subset \mathbb{R}^n$ eine Untermannigfaltigkeit des $\mathbb{R}^n$ mit Rand der Klasse C^1 sei.

Zudem sei $1 \leq p < \infty$ Dann existiert genaue eine stetige lineare Abbildung

$$S : W^{1,p}(\Omega) \to L^p(\partial\Omega)$$

mit der Eigenschaft $S(u) = u|_{\partial\Omega}$ für alle $u \in C^1(\bar{\Omega}) \cap W^{1,p}(\Omega)$

S nennt man Spur oder Spuroperator.

2.4 Diskretisierung mittels des Finiten-Volumen-Verfahrens

Um die Integrale mithilfe numerischer Methoden approximieren zu können, müssen zunächst die Integrale räumlich diskretisiert werden. Dabei gibt es hauptsächlich drei Kategorien, in die man diese Diskretisierungsverfahren aufteilen kann: die Finiten-Differenzen-Verfahren, Finite-Volumen-Verfahren und Finite-Elemente-Verfahren. In der Arbeit wird das Finite-Volumen-Verfahren verwendet, welches durch seine einfache Anwendbarkeit auf sowohl strukturierte als auch unstrukturierte Netze häufig verwendet wird. Die anderen beiden Diskretisierungsmethoden sind z.B. in [3] Kapitel 3.1.1. und 3.1.3. beschrieben. Dort findet sich auch in 3.1.4. eine Übersicht weiterer, aber weniger geläufiger Verfahren.
Unabhängig vom Diskretisierungsverfahren teilt man ein Gebiet, in dem man die Strömung untersuchen möchte, in eine endliche Anzahl geometrischer Elemente auf. Dabei ist es wichtig, dass sich die Elemente nicht überschneiden und zusammen das gesamte Gebiet abdecken. Bei zweidimensionalen Problemen werden meist Drei- oder Vierecke verwendet. Im dreidimensionalen Raum verwendet man meist Prismen, Pyramiden, Hexaeder oder Tetraeder. Genauer kann man dies mit folgenden Definitionen ausdrücken.

Definition 2.4.1. Zerlegung eines abgeschlossenen Gebietes Sei $D \subset \mathbb{R}^m$ ein abgeschlossenes Gebiet. Angenommen es existiert eine endliche Menge von offenen Gebieten $\{D_i\}_{i=1,\cdots N_{elem}}$ mit $N_{elem} \in \mathbb{N}$, $D_i \subset \mathbb{R}^m$, $D_i \neq \emptyset$ welche

D überdeckt, so dass gilt

$$D_i \subset D, \qquad \overline{D} = \bigcup_{i=1}^{N_{elem}} \overline{D}_i, \qquad D_i \cap D_j = \emptyset, \quad i \neq j$$

Dann heißt die Menge

$$M := \{D_i \,|\, i = 1, \cdots N_{elem}\}$$

eine Zerlegung des Gebietes D.

Definition 2.4.2. Sei $a \in \mathbb{R}^m$ und $b \in \mathbb{R}$. Dann heißt die Menge alle Punkte $x \in \mathbb{R}^m$, welche die Bedingung $\langle a, x \rangle \leq b$ erfüllen, Halbraum. Der Abschluss des Schnitts endlich vieler Halbräume heißt Polytop.

Ein Polytop im zweidimensionalen Raum nennt man auch Polygon und im dreidimensionalen Raum einen Polyeder.

Definition 2.4.3. Sei $D \subset \mathbb{R}^m$ mit $m \in \{2, 3\}$, D sei abgeschlossen und M eine Zerlegung von D. Außerdem seien D_i mit $i = 1, \cdots, N_{elem}$ Polytope. Dann heißt M finites Volumen Netz oder Triangulation wenn für $i \neq j$ eine der folgenden Bedingungen gilt.

- $\overline{D_j} \cap \overline{D_i} = \emptyset$
- $\overline{D_j} \cap \overline{D_i} \neq \emptyset$ und eine der folgenden Bedingungen gilt
  - $\overline{D_j}$ und $\overline{D_i}$ teilen sich genau eine Ecke
  - $\overline{D_j}$ und $\overline{D_i}$ teilen sich genau eine Kante
  - $\overline{D_j}$ und $\overline{D_i}$ teilen sich genau eine Seite (nur für $m = 3$)

Die D_i werden dann auch Netzzellen genannt.
Die genaue Ausprägung der Netze kann verschiedene Formen annehmen. Zum einen gibt es kartesische Netze, bei denen die Kanten der Netzzellen parallel zu den Achsen des kartesischen Koordinatensystems erzeugt werden. Andererseits gibt es den randangepassten Ansatz, bei dem die Kanten der Netzzellen an die tatsächliche Form des Randes angepasst. Dies ist aufwändiger

als der erste Ansatz, aber dafür können die Integrale besser über die Ränder aufgelöst werden. Aus diesem Grund wird das randangepasste Gitter auch meist in den Anwendungen bevorzugt. Eine schmematische Abbildung beider Varianten ist in 2.4 dargestellt

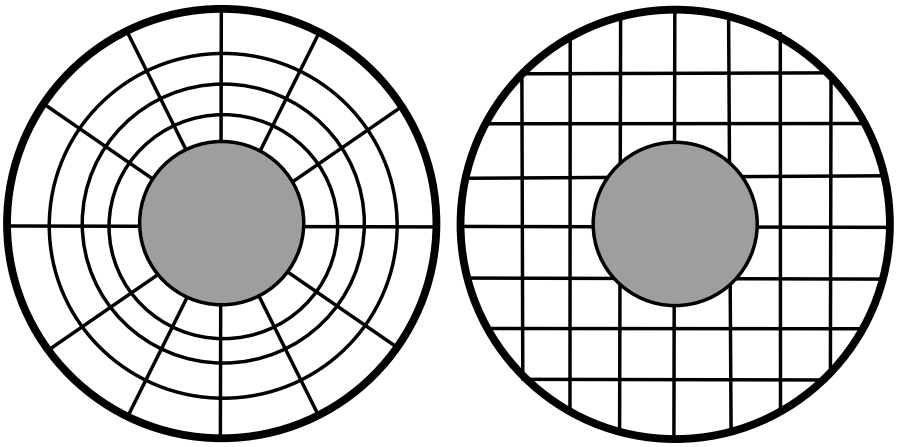


Abbildung 2.1: schematischer Vergleich eines randangepassten (links) und eines kartesischen(rechts) Gitters

Eine weitere Unterscheidung wird in der Struktur der Netze gemacht. Einerseits gibt es strukturierte Netze. Dabei erhält jede Zelle im dreidimensionalen Fall einen eindeutigen Index in x-, y- und z-Richtung und die Eck- bzw. Mittelpunkte können anhand der Koordinate $x_{i,j,k}, y_{i,j,k}$ und $z_{i,j,k}$ gefunden werden. Die Erzeugung dieser Netze ist vergleichsweise aufwändig und man kann nur Quader (bzw. im 2D-Fall Rechtecke) verwenden.
Eine andere Möglichkeit ist die Verwendung von unstrukturierten Netzen. Diesen fehlen die Indices, welche in Kombination jedes Element eindeutig kennzeichnen. Dadurch können auch Strukturen innerhalb des Netzes, beispielsweise die Identifizierung von Nachbarzellen, nicht mehr so einfach erkannt werden. Dafür sind sie in der Erzeugung einfacher und man kann beliebige geometrische Formen als Netzzellen verwenden (auch in Kombination innerhalb eines Netzes).

Bei der Diskretisierung mithilfe des Finiten-Volumen-Verfahrens werden die Integrale der Euler (Gleichung 2.17) bzw. Navier-Stokes-Gleichungen (Gleichung 2.12) mittels Kontrollvolumina approximiert. Dabei gibt es zwei Arten zu den Kontrollvolumina zu gelangen.
Eine Möglichkeit ist es, dass man diese analog zu den Zellen des Finiten-Volumen-Netzes wählt. Dabei befinden sich die Strömungsgrößen im Mittelpunkt der Netzzellen (auch „cell-centred“ genannt). Bei der anderen Variante wird ein sogenanntes Dualnetz erzeugt, welches aus einem neuen Netz besteht, in dem sich die Mittelpunkte an den ursprünglichen Eckpunkten der Netzzellen befinden. Hierdurch befinden sich die Strömungsgrößen an einem Eckpunkt der Zelle, deshalb spricht man auch von „cell-vertex“.
Für jedes dieser Kontrollvolumen Ω_i müssen dann wie im gesamten Gebiet Ω die Navier-Stokes bzw. Euler-Gleichungen gelten. Das Oberflächenintegral wird dabei mit Hilfe der Summe der Flüsse, die durch die Seitenflächen strömen, berechnet.
Die Diskretisierung der verschiedenen Teile der Navier-Stokes-Gleichungen soll nun kurz vorgestellt werden. Der Konvektionsanteil, welcher auch bei den Euler-Gleichungen auftritt, wird mithilfe eines upwind-Schemata diskretisiert. Diese basieren auf den physikalischen Eigenschaften der Euler-Gleichungen und betrachten den Einfluss stromaufwärts und stromabwärts getrennt.
Ein Vertreter dieser upwind-Schemata sind die Riemann-Löser. Dieses Problem tritt bei Unstetigkeiten an der Grenzfläche zwischen zwei Kontrollvolumen auf, die zum Beispiele durch Stöße entstehen können. Die Verwendung von eines solchen Schemas führt daher zu einer stabileren Diskretisierung des Konvektionsanteils.
Der hier verwendete Riemann-Löser ist das Roe-Schema, wie in [35] beschrieben. Das Roe-Schema ist dabei das heutzutage sehr beliebt, da es einfach sowohl auf strukturierten und unstrukturierten Netzen implementierbar ist und eine gute Auflösung der Randschichten besitzt. Dabei wird mittels einen zentralen Differenzen-Schemas und einer matrixwertigen künstlichen Viskosität approximiert. Außerdem werden noch ein Druckschalter (um Stöße erfassen zu können) und ein Zellstreckungskoeffizient (um stark gestreckte Teile des

Netzes besser behandeln zu können) eingeführt. Somit ergibt sich für ein Kontrollvolumen Ω_i folgende Approximation.

$$\oint_{\partial\Omega_i} \overrightarrow{F}_C \,\mathrm{d}S \approx \sum_{j\in\mathcal{N}(i)} \frac{1}{2}\left((\overrightarrow{F}_C^{ij})(W_i) + (\overrightarrow{F}_C^{ij})(W_j)\right)$$
$$- \frac{1}{2}|B_{ij}^{Roe}|\left\{\Psi_{ij}(W_j - W_i) - \xi s_{ij}(W)(1-\Psi_{ij})(L_j(W) - L_i(W))\right\}$$
$$L_i(W) = \sum_{j\in\mathcal{N}(i)} (W_j - W_i), \qquad |B_{ij}^{Roe}| = \frac{\partial(\overrightarrow{F}_C^{ij})}{\partial W}[W_{Roe}]$$
$$\Psi_{ij} = min\{\epsilon_\Psi \Psi_{ij}^\star, 1\} \qquad \Psi_{ij}^\star = \frac{(p_j - p_i)^2}{(p_j + p_i)^2} \qquad \epsilon_\Psi = 8$$
$$\lambda_i = \sum_{j\in\mathcal{N}(i)} \lambda_{ij} \qquad \lambda_{ij} = |v_{ij}^n| + A_{ij}c_{S,ij}$$
$$s_{ij}(W) = 1 + 2\frac{\sqrt{z_i}\sqrt{z_j}}{\sqrt{z_i} + \sqrt{z_j}} \qquad z_i = \frac{\lambda_i - \lambda_{ij}}{\lambda_{ij}}$$

wobei $\mathcal{N}(i)$ die Menge der Nachbarn der Zelle i und e_{ij} die dazugehörige Seitenfläche zwischen den Zellen i und j ist. Zudem sind A_{ij}, die Oberfläche der Seitenfläche, und die zugehörige Normale n_{ij} durch folgende Gleichungen gegeben.

$$A_{ij} = A(e_{ij}) = \int_{\Omega_i \cup \Omega_j} 1 \,\mathrm{d}s, \qquad \overrightarrow{n}_{ij} = A_{ij}(n_{x,ij}, n_{y,ij}, n_{z,ij})$$

Dabei sei $(n_{x,ij}, n_{y,ij}, n_{z,ij})$ der Einheitsnormalenvektor von e_{ij} Außerdem ist die Schallgeschwindigkeit $c_{S,ij}$, welche sich durch sich in den angrenzenden Zellen unterscheidende Dichten auch verändern kann, als Mittelwert der Schallgeschwindigkeit der angrenzenden Zellen gegeben. Die Normalengeschwindigkeit v_{ij}^n ist das Skalarprodukt der gemittelten Geschwindigkeit der beiden angrenzenden Zellen und der Normale, also

$$v_{ij}^n = (0,5(\overrightarrow{v}_i + \overrightarrow{v}_j) \cdot \overrightarrow{n}_{ij})$$

W_{Roe} steht für die nach Roe [36] gemittelten Werte von $\overrightarrow{W}$ aus 2.17. Diese

werden mithilfe von

$$\overline{s} = \frac{\sqrt{\rho_i} s_i + \sqrt{\rho_j} s_j}{\sqrt{\rho_i} + \sqrt{\rho_j}}$$

berechnet, wobei der Index für die zugehörige Zelle steht und s eine drei Geschwindigkeitskomponenten u, v, w oder die spezifische Totalenthalpie H annehmen kann. Für die Dichte verwendet man

$$\overline{\rho} = \sqrt{\rho_i \rho_j}$$

Die Diskretisierung im Fall des viskosen Flusses wird analog auf den Kontrollvolumen ausgeführt. Dabei werden die Geschwindigkeitskomponenten, die Viskosität η und der Wärmeleitungskoeffizient k der Seitenfläche e_{ij} durch Mittelung der Werte der angrenzenden Zellen berechnet. Für die 1.Ableitungen wird die Green-Gauß-Methode verwendet. Sie verwendet die Approximation

$$\text{grad } U \approx \frac{1}{\Omega} \int_{\partial\Omega} U \overrightarrow{n} \, \mathrm{d}S$$

für eine beliebige Größe U. Somit ergibt sich beispielsweise für T in einer Zelle i

$$\nabla T_i \approx \frac{1}{\Omega_i} \int_{\partial\Omega_i} T \overrightarrow{n} \, \mathrm{d}S \approx \frac{1}{\Omega_i} \sum_{j \in \mathcal{N}(i)} A_{ij} \frac{n_{ij}}{2} (T_i + T_j)$$

Analog kann dies für die Geschwindigkeitskomponenten u, v, w berechnet werden. Der Gradient auf der Seitenfläche e_{ij} wird als Durchschnitt der Gradienten der angrenzenden Zellen berechnet.
Mithilfe dieser approximierten Ableitungen kann man dann den viskosen Teil durch

$$\oint_{\partial\Omega_i} \overrightarrow{F}_V \, \mathrm{d}S \approx \sum_{j \in \mathcal{N}(i)} (\overrightarrow{F}_V^{ij}) \quad ,$$

d.h. als Summe über die approximierten Variablen $\overrightarrow{F}_V$ aller Seitenflächen, berechnen.
Die Navier-Stokes-Gleichungen können auch als diskretisierte Differentialgleichung umgeschrieben werden und haben dann die kompaktere Form

$$\frac{d}{dt} \overrightarrow{W} = -M^{-1} R(\overrightarrow{W}) \tag{2.19}$$

wobei M für die Massenmatrix $M = diag(diag(V_i)) \in \mathbb{R}^{5N \times 5N}$, V_i für das Volumen der Zelle D_i und R für das Residuum der Variable $\overrightarrow{W}$ steht. (siehe [25, S.104ff.])
Zum Schluss soll noch einmal auf die im Abschnitt 2.1 vorgestellten Anfangs- und Randbedingungen eingegangen werden. Zur Implementierung dieser werden sogenannte Dummy-Zellen benötigt. Dafür wird bei der Erstellung der Netzstruktur eine weitere Schicht Zellen außerhalb des eigentlichen Gebiets generiert. Der Zweck dieser Zellen ist, dass man mit ihnen die Flüsse, Gradienten usw. entlang der Ränder besser berechnen kann. Dafür werden die Randbedingungen durch die Wahl der Werte für diese Dummy-Zellen abgebildet. Wie genau diese Werte für die Dummy-Zellen aussehen hängt dabei aber auch davon ab, ob man den cell-centred oder cell-vertex Ansatz wählt und ob man auf einem strukturierten oder unstrukturierten Netz arbeitet. Hier sollen kurz die die feste Wand, welches durch den Teil des Flugzeuges und die farfield- Bedingung am äußeren Rand eingegangen werden. Das Netz hat die Form eines unstrukturierten cell-centred Ansatzes.
Im Fall der Navier-Stokes-Gleichungen erhalten die Dummy-Zellen bezüglich der Dichte und der spezifischen Energie die Werte der Randzelle. Für die Geschwindigkeit gilt

$$\overrightarrow{v}_{Dummy} = -\overrightarrow{v}_{Rand}$$

Bei den Euler-Gleichungen nimmt man für den Druck und die Dichte der Dummy-Zelle die der Randzelle an. Der Geschwindigkeitsvektor der Dummy-Zelle entspricht

$$\overrightarrow{v}_{Dummy} = \overrightarrow{v}_{Rand} - 2 < \overrightarrow{v}_{Rand}, \overrightarrow{n}_{Rand,Dummy} > \overrightarrow{n}_{Rand,Dummy}$$

wobei $\overrightarrow{n}_{Rand,Dummy}$ die Normale an der Grenzfläche zwischen Randzelle und korrespondierender Dummy-Zelle darstellt. Die Fernfeld-Bedingung wird mithilfe des Vektors wie in 2.16 beschrieben dargestellt, wobei $\overrightarrow{W}_a$ die Variable $\overrightarrow{W}$ in der Dummy-Zelle darstellt.

Kapitel 3

Mehrgitter-Verfahren

In diesem Kapitel soll die Lösungsmethode, welche für die in vorherigen Kapitel vorgestellten Navier-Stokes- bzw. Euler-Gleichungen verwendet wird, vorgestellt werden. Beim sogenannten Mehrgitterverfahren arbeitet man nicht nur auf einem Netz, stattdessen wird dessen Aufteilung wird sukzessive vergröbert sodass man mit einer Sequenz von systematisch vergröberten Netzen rechnet. Dabei werden die berechneten Lösungen mithilfe der Prolongation und Restriktion von einem Gitter auf ein anderes übertragen. Mithilfe dieses Verfahrens kann eine stärkere Konvergenz als mit dem Lösen auf einem Gitter erreicht werden. Man betrachtet das Beispiel des zweidimensionalen diskreten Poisson-Problem mit Dirichlet-Randbedingungen, also

$$\begin{aligned} -\Delta_h u_h &= f_h \,\text{auf}\ \Omega_h \\ u &= g \,\text{auf}\ \partial\Omega_h \end{aligned} \tag{3.1}$$

mit $\Delta f = \text{div}(\text{grad}\ f)$ als Laplace-Operator. In der folgenden Tabelle 3 sind verschiedene Methoden und ihre Konvergenzgeschwindigkeit für die zweidimensionale Poisson-Gleichung. Man erkennt, dass das Mehrgitterverfahren ein ähnlich gutes Konvergenzverhalten besitzt wie auf dieses Problem angepasste Löser und eine deutlich bessere als andere allgemeine iterative Verfahren.

Tabelle 3.1: Vergleich verschiedener Löser bezüglich ihrer Konvergenzgeschwindigkeit für ein 2D-Poisson-Problem, siehe [41], S.14

Verfahren	Konvergenzgeschwindigkeit
Jacobi-Verfahren	$\mathcal{O}(N^2 \log \varepsilon)$
Gauß-Seidel-Verfahren	$\mathcal{O}(N^2 \log \varepsilon)$
implizite Methode der alternierenden Richtungen	$\mathcal{O}(N \log N \log \varepsilon)$
Buneman Algorithmus	$\mathcal{O}(N \log N)$
Mehrgitterverfahren	$\mathcal{O}(N \log \varepsilon)$

wobei N für die Gesamtanzahl der Unbekannten und ε für die Abbruchgenauigkeit steht.
Da sowohl die Navier-Stokes- als auch den Euler-Gleichungen von nichtlinearer Art sind, wird nicht ein klassische Mehrgitterverfahren, sondern ein sogenannte FAS („Full Approximation Scheme“) verwendet. Hierbei wird das klassische Mehrgitterverfahren für lineare Probleme auf ein nichtlineares Problem verallgemeinert. Dabei wird aber formal nicht wie sonst üblich mit der Glättung der Fehler gearbeitet, sondern mit den kompletten Approximationen der diskreten Lösung auf dem Grobnetz. Führt man aber ein FAS auf ein lineares Problem aus, so erhält man wieder ein klassisches Mehrgitterverfahren.

3.1 Motivation des Mehrgitter-Verfahrens

Möchte man nun das in 2.4 beschriebene diskrete Problem mittels numerischer Methoden approximieren, so steht man zunächst vor einer großen Anzahl an möglichen Verfahren. Eine Schwierigkeit, die bei den Euler- und Navier-Stokes-Gleichungen besteht, ist die Nichtlinearität dieser Gleichungssysteme. Dies schränkt die möglichen Methoden schon stark ein. Ein weiteres Problem ergibt sich daraus, dass die in der relevanten Anwendungsfälle auf Netzen mit einer großen Anzahl an Gitterpunkten (und dementsprechend auch einer hohen Anzahl an Freiheitsgraden, für die das Problem gelöst werden soll) und einer hohen Genauigkeit benötigt werden. Deshalb sucht man ein Verfahren, welches möglichst linear mit der Anzahl der Freiheitsgrade

wachsen soll. Eine Möglichkeit ist das Mehrgitterverfahren. Dieses wurde ursprünglich für diskrete Randwertproblem auf dem Einheitsquadrat von Fedorenko [9] bzw. für die Poisson-Gleichung von Bakhvalov [2] entwickelt. Diese Verfahren wurden dann von Brandt weiterentwickelt, sodass sie auch für nichtlineare Probleme verwendet werden können.
Im Folgenden wird die Grundidee des Mehrgitterverfahrens mithilfe eines Beispiels aus [21], S.65 vorgestellt werden.
Sei das Randwertproblem

$$\begin{aligned} -u''(x) &= 0, \qquad x \in (0,1) \\ u(0) = u(1) &= 0 \end{aligned}$$

Dieses Problem wird nun mit n=64 diskretisiert. Dadurch erhält man das LGS $A\overrightarrow{u} = \overrightarrow{b}$ mit

$$A = \begin{pmatrix} 2 & -1 & 0 & \cdots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 2 & -1 \\ 0 & \cdots & 0 & -1 & 2 \end{pmatrix}, \quad \overrightarrow{u} = \begin{pmatrix} \frac{1}{n} \\ \vdots \\ \vdots \\ \vdots \\ \frac{n-1}{n} \end{pmatrix}, \quad \overrightarrow{b} = \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ \vdots \\ 0 \end{pmatrix}$$

mit $A \in \mathbb{R}^{n-1 \times n-1}$, $u \in \mathbb{R}^{n-1}$, $b \in \mathbb{R}^{n-1}$. Als Startvektoren $\overrightarrow{u}^{(0)(k)}$ werden Vektoren mit unterschiedlich vielen Vorzeichenwechseln, also typischerweise Sinus- oder Cosinus-Funktionen gewählt.

$$\overrightarrow{u}^{(0)}(s) = \begin{pmatrix} \sin(\frac{1}{n}s\pi) \\ \vdots \\ \sin(\frac{j}{n}s\pi) \\ \vdots \\ \sin(\frac{(n-1)}{n}s\pi) \end{pmatrix}$$

mit $j \in \{1, \ldots n-1\}$. Das Gauss-Seidel-Verfahren zur näherungsweisen Lösung des linearen Gleichungssystems ist ein Iterationsverfahren und hat all-

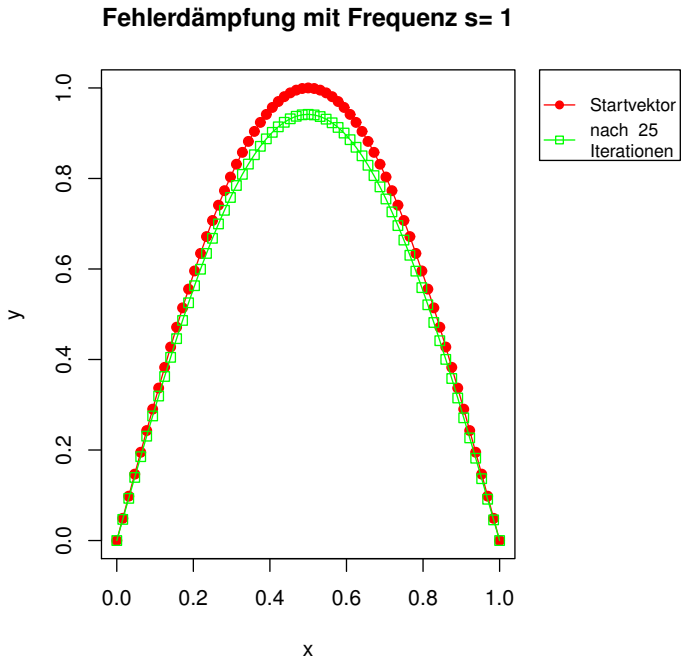
gemein die Form

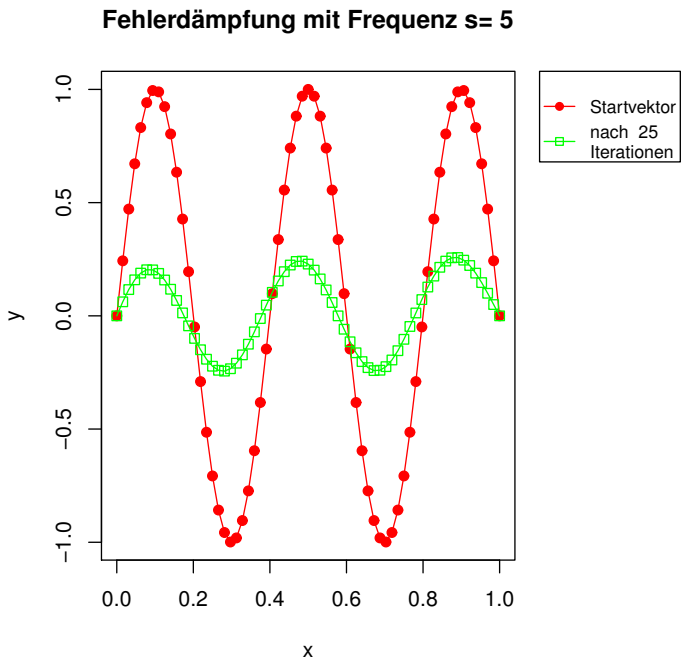
$$u_k^{(m+1)} = \frac{1}{a_{kk}} \left(b_k - \sum_{i=1}^{k-1} a_{ki} u_i^{(m+1)} - \sum_{i=k+1}^{n-1} a_{ki} u_i^{(m)} \right) \tag{3.2}$$

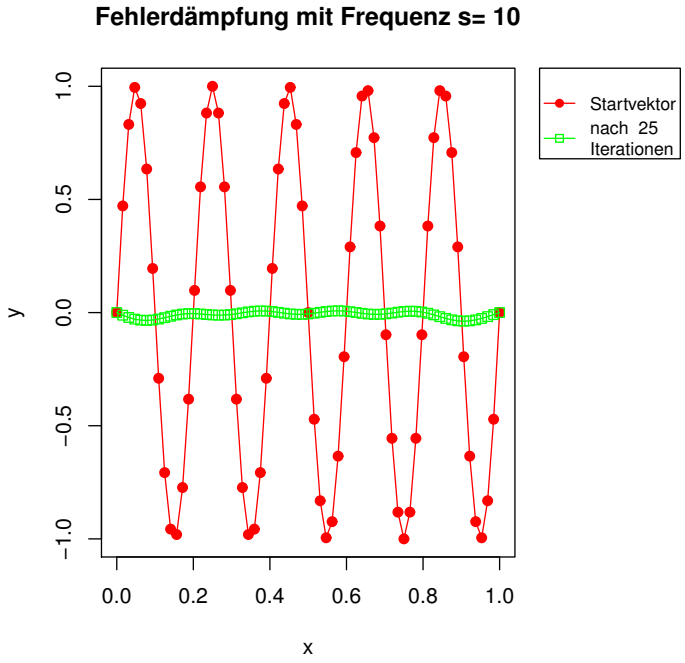
für $k \in \{1, \ldots, n-1\}$ Im Fall des hier angegebenen Problems erhält man

$$\begin{aligned} u_1^{(m+1)} &= \frac{1}{2} \left(u_2^{(m)} \right) \\ u_k^{(m+1)} &= \frac{1}{2} \left(u_{k-1}^{(m+1)} + u_{k+1}^{(m)} \right) \\ u_{n-1}^{(m+1)} &= \frac{1}{2} \left(u_{n-2}^{(m+1)} \right) \end{aligned}$$

Dieses Verfahren wurde 25-mal hintereinander in R für $s = 1$, $s = 5$ und $s = 10$ ausgeführt.




Fehlerdämpfung mit Frequenz s= 5
Startvektor
nach 25
Iterationen
y
x
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0


Fehlerdämpfung mit Frequenz s= 10
Startvektor
nach 25
Iterationen
y
x
1.0
0.5
0.0
−0.5
−1.0
0.0
0.2
0.4
0.6
0.8
1.0

Dabei sieht man, dass es zu einer unterschiedlich starken Glättung kommt. Die Startvektoren mit einer niedrigen Frequenz werden deutlich schwächer gedämpft als die mit einer hohen Frequenz. Dieses Verhalten macht sich auch beim Mehrgitterverfahren zu Nutze. Hierbei führt man ein solches iteratives Verfahren auf den Fehler $e^{(m)} = u^{(m)} - u$ aus. Besteht dieser aus Komponenten verschiedener Frequenz, so wird der Fehler nicht unbedingt kleiner wird durch das Verfahren, aber der Fehler wird geglättet. Ein solches Verfahren soll im Folgenden Glätter genannt werden.
Das Residuum $r^{(m)} = b - Ax^{(m)} = Ae^{(m)}$ hängt vom Fehler $e^{(m)}$ ab. Da die hochfrequenten Anteile des Fehlers nach einigen Iterationen geglättet wurden, hängt das Residuum größtenteils nur noch von den niedrigfrequenten Anteilen ab. Durch diese Glättung nimmt man an, dass man das Residuum nun auch auf einem gröberen Gitter (meist mit doppelter Schrittweite $2h$) approximieren kann. Die Korrektur der Näherung $u^{(m)}$ wäre dann auf dem neuen Gitter aufgrund seiner geringeren Dimension auch mit weniger Aufwand zu berechnen.
Ein weiterer Vorteil der groben Netze ist, dass hier die glatten Komponenten des Fehlers des feinen Gittern wieder hochfrequent sind und daher erneut mit einem geeigneten Operator geglättet werden kann (siehe [19], S.13). Beim Mehrgitterverfahren wird daher der Fehler sukzessive auf immer gröber werdenden Gittern geglättet, bis am Ende auf dem gröbsten Gitter eine Art Defektgleichung approximativ durch erneutes Anwenden des Glätters gelöst wird. Die wichtigsten Bestandteile des Verfahrens sind daher die Interpolations- und Projektionsoperatoren zum Transfer des Fehlers von einem Gitter auf ein anderes und die Glättungsoperation.

3.2 Netze

In diesem Kapitel soll noch einmal auf die Netze eingegangen werden. Dabei soll die in Abschnitt 2.4 vorgestellte Definition 2.4.3 der finiten Volumen Netze gelten. Im Mehrgitterverfahren benötigt man aber nicht nur eines dieser Netz, sondern, dem Namen gerecht, rechnet man auf mehreren Netzen. Dabei

wird mithilfe der folgenden Definition eine Vergröberung bzw. Verfeinerung des Gitters dargelegt.

Definition 3.2.1. Seien $M = \{D_i | i = 1, \cdots N_{elem}\}$ und $M' = \{D'_i | i = 1, \cdots N'_{elem}\}$ beiden finite Volumen Netze auf dem abgeschlossenen Gebiet $D \subset \mathbb{R}^n$. M' heißt dann Verfeinerung von M, wenn für alle $D_i \in M$, $i = 1, \cdots N_{elem}$ eine Menge von Indices $C(i) = \{i_1, \cdots, i_N\}$ existiert, sodass gilt

$$J_i = \{D'_{ik} | k = 1, \cdots N\}$$

dass J_i ein finites Volumen Netz auf D_i ist. Man schreibt auch $M \subset M'$. Ein Gebiet D'_{ik}, $k \in C(i)$ sei eine Unterzelle von D_i.

Bei Standardverfahren zur Vergröberung des Netzes wird in jede Richtung mit dem Faktor 2 vergröbert. Somit enthält im zweidimensionalen Fall eine neue Zelle des nächstgröberen Gitters vier ursprüngliche Zellen und im dreidimensionalen acht. Sei ein Netz in 2D mit 64 Elementen, also Netzgröße $h = \frac{1}{8}$ wenn die Skala in beide Richtungen von 0 bis 1 geht, wie in 3.1 als cell-centred Ansatz gegeben.
Die verbreitesten Formen der Vergröberung sind die Standardvergröberung(2h), die Verdoppelung nur in eine Richtung, auch „semi-coarsening“ genannt und die Vergröberung um den Faktor 4 in beide Richtungen. Diese Varianten sind in 3.2, 3.3, 3.4 für einen cell-centred Ansatz gezeigt.

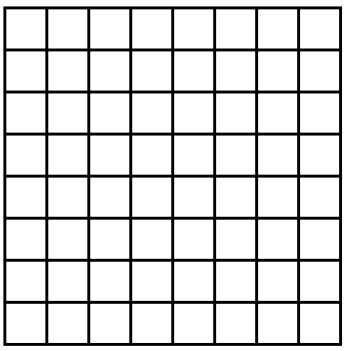


Abbildung 3.1: 2D-Netz mit 64 Elementen

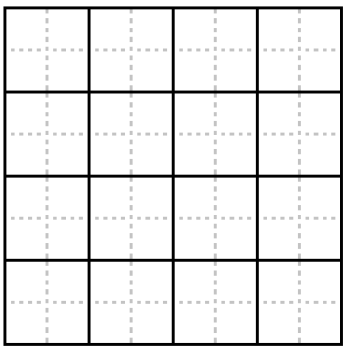


Abbildung 3.2: Standardvergrö-berung 2h

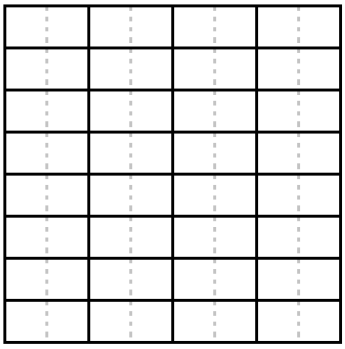


Abbildung 3.3: Vergröberung nur in eine Richtung mit 2h(„semi-coarsening“)

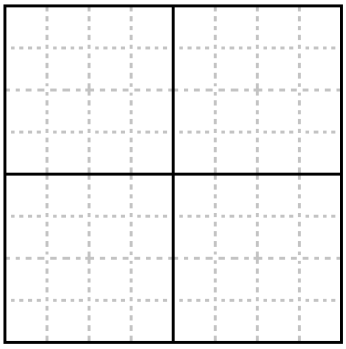


Abbildung 3.4: Vergröberung 4h

Wie genau man zu den vergröberten Gittern gelangt, hängt dabei von der verwendeten Struktur des Netzes ab. In strukturierten Gittern kann man die Nachbarn anhand der Indices schnell identifizieren und so die neuen Zellen zusammenfassen. Bei den unstrukturierten Netzen muss stattdessen ein Agglomerationsalgorithmus verwendet werden. Ein Vertreter dieser Algorithmen ist der MGridGen-Algorithmus [31], welcher mittels gewichteter Graphen eine Sequenz von Grobnetzen erzeugt. Dieses Verfahren wird ebenso wir das neue, sich an den strukturierten Netzen orientierte, Verfahren im Abschnitt 4.2 erläutert.
Mithilfe dieser Verfahren erhält man eine Sequenz von Gittern $M_1, \cdots M_n$ mit $M_n \subset \cdots \subset M_1$.
Die oben beschriebene Einbettung des groben Gitters in das feine wird fast ausschließlich verwendet, ist aber nicht zwingend notwendig. Es erleichtert das Transferieren zwischen den Gittern und ermöglicht die Angabe von allgemeinen Transferoperatoren.
Die in der Praxis verwendeten Netze sind meist unstrukturierter Art. Das liegt daran, dass zur Netzgenerierung Programme verwendet werden, die diese automatisch unstrukturierte Netze erzeugen, während bei strukturierten Netzen meist noch eine Nachbearbeitung benötigt wird. Wie an dem Beispiel aus der Einleitung jedoch gezeigt, gibt es auch Fälle in denen die strukturierten Netze in Verbindung mit einem Mehrgitterverfahren eine geringere

Konvergenz als unstrukturierte Netze besitzen. Dies ist auch die Motivation dieser Arbeit. Kann man mithilfe einer Agglomerationsstragie, ähnlich der in strukturierten Netzen, und der Verwendung von Interpolations- und Projektionsoperatoren aus strukturierten Netzen eine Leistungssteigerung in unstrukturierten Netzen messen?
Zudem gibt es auch Netze, die sowohl aus strukturierten Netzteilen als auch unstrukturierten Netzteilen bestehen. Ein Beispiel für ein solches Netz ist in 3.5 abgebildet.

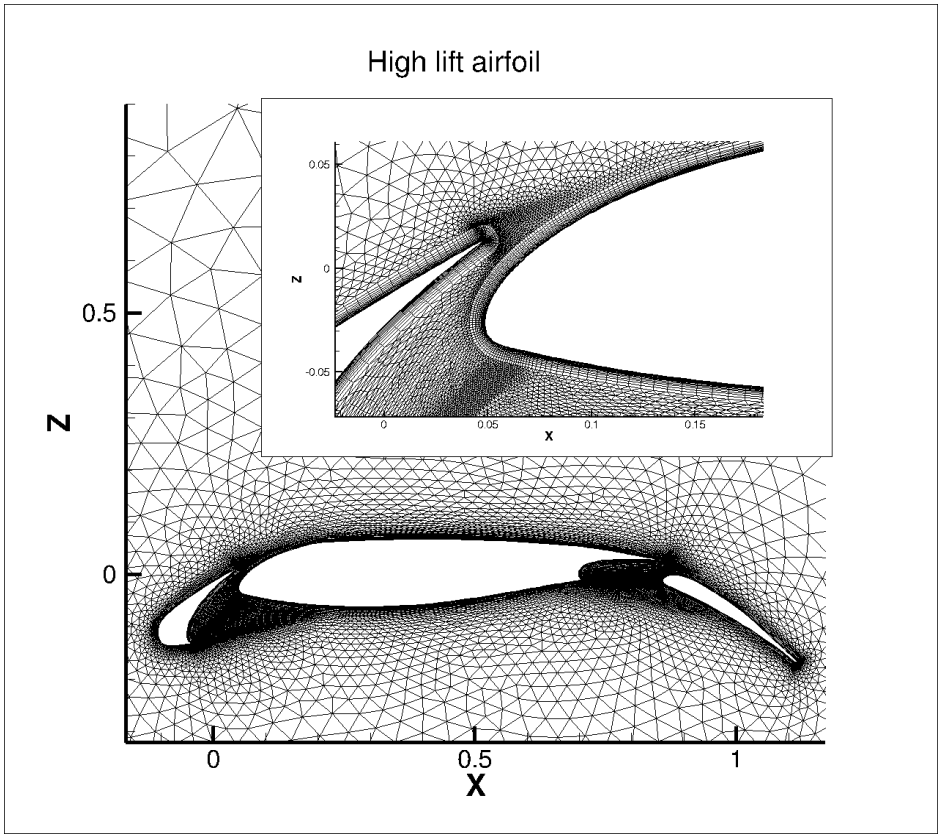


Abbildung 3.5: Netz mit strukturierten und unstrukturierten Anteilen

Hierbei ist am innerer Rand ein strukturiertes Netz gegeben und bei dem restlichen Teil des Netzes handelt es sich um ein unstrukturiertes. Hier stellt sich die Frage wie man den strukturierten Teil identifiziert und dann weiter mit ihm vorgeht.

3.3 FAS

Das klassische Mehrgitterverfahren ist eine numerische Lösungsmethode für lineare Probleme. Die in Abschnitt 2.12 und 2.17 beschriebenen Probleme sind aber nichtlineare Probleme. Um das Mehrgitterverfahren auf diese anwenden zu können, müssen also einige Anpassungen vorgenommen werden. Grundsätzlich gibt es 2 Möglichkeiten das Verfahren auf nichtlineare Probleme anzuwenden. Zum einen kann man das Problem linearisieren, in dem man beispielsweise ein Newton-Verfahren ausführt und dann für jede Iteration ein Mehrgitterverfahren zum Lösen dieser anwendet. Zum anderen kann man das Mehrgitterverfahren verallgemeinern um es auch für nichtlineare Probleme anwenden zu können. Das dabei entstehende Verfahren nennt man auch FAS („Full Approximation Scheme“), welches als erstes von Brandt [5] beschrieben wurde.
Der Ablauf des FAS ist dabei nach [41, S.157] wie folgt. Sei zunächst ein nichtlineares Problem

$$Nu = f \tag{3.3}$$

gegeben. Dieses wird dann auf einem Gitter M_1 mit Schrittweite h diskretisiert. Diese diskrete, nichtlineare Problem sei durch

$$N_1 u_1 = f_1$$

gegeben. Das Residuum $R_h^{(m)}$ ist durch

$$R_1^{(m)} = f_1 - N_1 u_1^{(m)}$$

gegeben, wobei $u_1^{(m)}$ die Näherung von u_1 in der Iteration m ist. Der Fehler $e_1^{(m)}$ ist durch

$$e_1^{(m)} = f_1 - N_1 u_1^{(m)}$$

gegeben Zudem sei ein Glätter S_1 gegeben. Wendet man diesen auf $u_1^{(m)}$ an, so erhält man die geglättete Näherung $\bar{u}_1^{(m)} = S_1(u_1^{(m)}, N_1, f_1)$ Desweiteren sei eine Diskretisierung auf einem gröberen Gitter M_2 mit Schrittweite H

und $H > h$ durch

$$N_2 u_2 = f_2$$

gegeben. Zudem wird die Größe $w_2^{(m)} = \bar{u}_2^{(m)} + \hat{v}_2^{(m)}$ benötigt, da auf dem Netz M_2 die Gleichung

$$N_2(\bar{u}_2^{(m)} + \hat{v}_2^{(m)}) = N_2 \bar{u}_2^{(m)} + R_2^{(m)}$$

gelöst werden muss. Außerdem benötigt man Transferoperatoren T_R und T_P. Der Restriktionsoperator T_R überträgt einen Vektor vom feinen auf das grobe Netz und der Prolongationsoperator überträgt einen Vektor vom groben auf das feine Netz.

$$T_R^1 : \mathbb{R}^{\sharp M_1} \to \mathbb{R}^{\sharp M_2}, \qquad T_P^2 : \mathbb{R}^{\sharp M_2} \to \mathbb{R}^{\sharp M_1}$$

Zusätzlich sei eine Näherung $u_h^{(m)}$ in der m-ten Iteration auf dem Netz M_1 gegeben. Um $u_h^{(m+1)}$ zu berechnen, geht man wie folgt vor.

Tabelle 3.2: Ablauf des FAS auf 2 Netzen

1. Zunächst wendet man S_1 auf $u_1^{(m)}$ an, sodass man die geglättete Näherung $\bar{u}_1^{(m)}$ erhält. Dies ist nun die neue Näherung mit der man arbeitet.

$$\bar{u}_1^{(m)} = S_1(u_1^{(m)}, N_1, f_1)$$

2. Danach berechnet man das Residuum $R_1^{(m)}$ mithilfe der neuen Näherung.

$$R_1^{(m)} = f_1 - N_1 \bar{u}_1^{(m)}$$

3. Im Anschluss überträgt man das Residuum mittels des Restriktionsoperators auf das gröbere Netz M_2

$$R_2^{(m)} = T_R^1 R_1^{(m)}$$

4. Zudem wird auch die Näherung auf M_2 übertragen.

$$\bar{u}_2^{(m)} = T_R^1 \bar{u}_1^{(m)}$$

5. Auf dem Netz M_2 wird nun die Gleichung $N_2(w_2^{(m)}) = R_2^{(m)} + N_2\bar{u}_2^{(m)}$ approximativ durch Anwenden des Glätters S_2 gelöst.

$$\bar{w}_2^{(m)} = S_2(w_2^{(m)}, N_2, R_2^{(m)} + N_2\bar{u}_2^{(m)})$$

6. Mit der Hilfsgröße $\bar{w}_2^{(m)}$ wird dann die Korrektur $\hat{v}_2^{(m)}$ berechnet

$$\hat{v}_2^{(m)} = \bar{w}_2^{(m)} - \bar{u}_2^{(m)}$$

7. Diese Lösung $\hat{v}_2^{(m)}$ wird mithilfe des Prolongationsoperators wieder auf das feine Netz übertragen

$$\hat{v}_1^{(m)} = T_P^2 \hat{v}_2^{(m)}$$

8. Die korrigierte Näherung wird dann aus der Summe der alten Näherung und der Korrektur berechnet.

$$u_1^{(m),nachher} = \hat{v}_1^{(m)} + \bar{u}_1^{(m)}$$

9. Im letzten Schritt wird die korrigierte Näherung erneut mittel S_1 geglättet.

$$u_1^{(m+1)} = S_1(u_1^{m,nachher}, N_1, f_1)$$

Handelt es sich bei N_1 und N_2 um lineare Operatoren, so gilt

$$\begin{aligned} N_2(\bar{u}_2^{(m)} + \hat{v}_2^{(m)}) &= r_2^{(m)} + N_2\bar{u}_2^{(m)} \\ N_2\bar{u}_2^{(m)} + N_2\hat{v}_2^{(m)} &= r_2^{(m)} + N_2\bar{u}_2^{(m)} \\ N_2\hat{v}_2^{(m)} &= r_2^{(m)} \end{aligned}$$

Somit muss man statt der Schritte 4 bis 6 nur noch $N_2\hat{v}_2^{(m)} = r_2^m$ approximativ auf M_2 lösen. Somit erhält man in diesem Fall ein klassisches Mehrgitterverfahren wie z.B. in [41], S. 39 beschrieben. Ein solches Mehrgitterverfahren auf zwei Netzen nennt man auch 2V-Zyklus.
Dieses Verfahren für 2 Gitter kann man auch auf eine beliebige Sequenz von Gittern erweitern. Die Idee dabei ist, dass man die Gleichung im 5.Schritt auf dem gröberen Netz nicht durch den Glätter approximiert, sondern durch erneutes Anwenden des 2V-Zyklus approximiert. Dabei arbeitet man aber statt auf dem feinen und dem groben Netz auf dem groben und einem noch gröberen Netz.

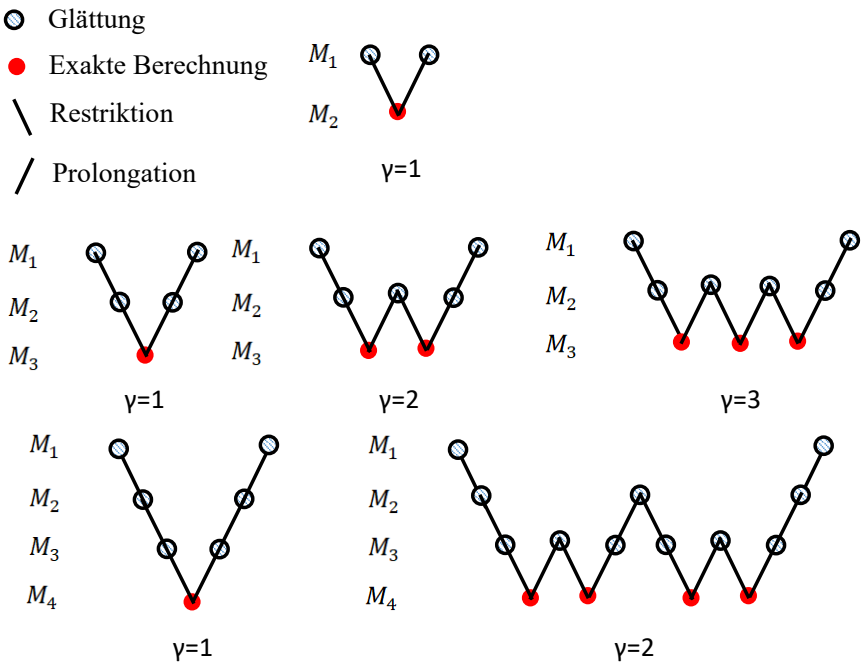


Abbildung 3.6: verschiedene Arten eines Mehrgitterzyklus

Die Anzahl, wie oft das Verfahrens mit zwei Netzen ausgeführt, wird dabei mit γ bezeichnet. Je nachdem erhält man dann verschiedene Mehrgitterzyklen, wie in Abbildung 3.6 dargestellt. Zyklen mit $\gamma = 1$ werden auch als V-Zyklen und mit $\gamma = 2$ als W-Zyklen bezeichnet. In der ersten Zeile befindet sich ein 2V-Zyklus, in der zweiten Zeile sind Beispiele für ein Mehrgit-

terverfahren auf 3 Netzen angegeben und in der letzten Zeile sind der 4V- bzw. 4W-Zyklus angegeben. Letzterer wird auch später in den Beispielen verwendet.
Allgemein wird ein Mehrgittterverfahren wie folgt definiert. Gegeben sei das Problem 3.3. Es sei eine Sequenz von Netzen $M_n \subset \cdots \subset M_1$ gegeben. Für jedes dieser Netze seien die Operatoren

$$N_k : \mathbb{R}^{\sharp M_k} \to \mathbb{R}^{\sharp M_k}, \qquad S_k : \mathbb{R}^{\sharp M_k} \to \mathbb{R}^{\sharp M_k}, \qquad T_P^{k+1} : \mathbb{R}^{\sharp M_{k+1}} \to \mathbb{R}^{\sharp M_k},$$
$$T_R^k : \mathbb{R}^{\sharp M_k} \to \mathbb{R}^{\sharp M_{k+1}}$$

gegeben. Zu einem Netz M_k ist N_k die Diskretisierung von N, S_k steht für den Glätter, I_P^k steht für den Prolongationsoperator und I_R^k steht für den Restriktionsoperator.
Ein Mehrgitterzyklus hat dann fast die gleiche Form wie in 3.2 beschrieben. Statt auf den Netzen M_1 und M_2 führt man den Mehrgitteralgorithmus auf M_k und M_{k+1} aus. Dabei verändern sich sich dementsprechend auch die Näherungen und Operatoren. Zudem werden anstelle von Schritt 5 folgende Schritte ausgeführt.

5a Man berechnet die rechte Seite

$$f_{k+1} = d_{k+1}^{(m)} + N_{k+1}\bar{u}_{k+1}^m$$

5b Ist $k = n$, so löst man die Gleichung

$$N_{k+1}w_{k+1}^{(m)} = d_{k+1}^{(m)} + N_{k+1}\bar{u}_{k+1}^{(m)}$$

näherungsweise durch

$$\bar{w}_{k+1}^{(m)} = S(w_{k+1}^{(m)}, N_{k+1}, d_{k+1}^{(m)} + N_{k+1}\bar{u}_{k+1}^{(m)})$$

Sonst führt man γ-mal das Mehrgitterverfahren aus, jedoch auf M_{k+1} und M_{k+2} statt M_k und M_{k+1}.

3.4 Operatoren

Die Operatoren dienen dem Übertragen der Variablen zwischen zwei benachbarten Gittern. Dabei ist der Prolongationsoperator I_P^{k+1} für den Übergang vom feinen auf ein grobes Gitter und der Restriktionsoperator I_R^k von einem feinen auf ein feines Gitter.

$$I_P^{k+1} : \mathbb{R}^{\sharp M_{k+1}} \to \mathbb{R}^{\sharp M_k}, \qquad I_R^k : \mathbb{R}^{\sharp M_k} \to \mathbb{R}^{\sharp M_{k+1}}$$

Eine mögliche Wahl für den Restriktionsoperator ist die volumengewichtete Interpolation und für den Prolongationsoperator die Injektion.
Sei D_i^{k+1} die i-te Zelle in Netz M_{k+1}. Alle Zellen aus M_k, für die gilt

$$\bigcup_j D_j^k = D_i^{k+1}$$

sei der Index j in Unterzelle(C_i^{k+1}). In Abbildung 3.7 sei links M_1 und rechts M_2 dargestellt. Dann gilt z.B. $\{17, 18, 25, 26\} \in$ Unterzelle(C_5^2)
Die Injektion für eine Variable U ist dann gegeben durch

$$U_j^k = U_i^{k+1}, \quad j \in \text{Unterzelle}(D_i^{k+1})$$

Für das Beispiel in Abbildung 3.7 wäre $U_1^1 = U_2^1 = U_0^1 = U_{10}^1 = U_1^2$
Die volumengewichtete Interpolation wird durch

$$U_i^{k+1} = \frac{1}{V_i^{k+1}} \sum_{j \in \text{Unterzelle}(D_i^{k+1})} V_j^k U_j^k \tag{3.4}$$

mit V_i^k ist das Volumen der Zelle D_i^k berechnet. Im Beispiel wäre dies also

$$U_2^2 = \frac{1}{V_2^2} \sum_{j \in \{3,4,11,12\}} V_j^1 U_j^1 \tag{3.5}$$

Eine Herleitung für diese Operatoren findet sich beispielsweise in [25, S.158 ff.]
Um das Residuum R^{k+1} einer agglomerierten Zelle D_i^{k+1} zu berechnen, summiert man alle Flüsse der zugehörigen Unterzellen auf. Da sich die Flüsse

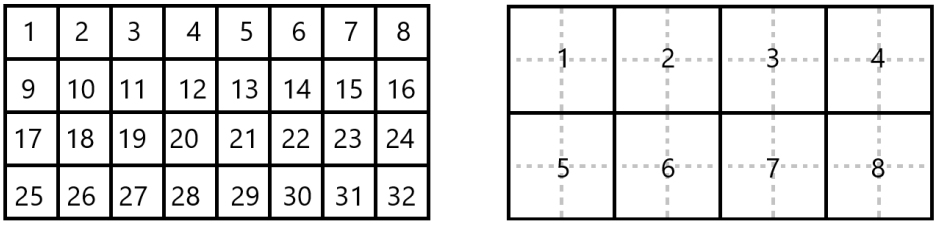


Abbildung 3.7: Beispiel eines feinen und groben Netzes

innerhalb der Zelle aber gegenseitig aufheben, muss man lediglich über die Kanten der agglomerierten Zelle summieren.
Diese beiden vorgestellten Operatoren benötigen nur die Informationen über die Zellen, die durch die Agglomeration zusammengehören und funktionieren somit sowohl in strukturierten als auch unstrukturierten Netzen. Durch die Verwendung von strukturierten Netzen ergibt sich eine große Möglichkeit weiterer Operatoren. Im folgenden sollen drei weitere in der Literatur zu findende Operatoren vorgestellt werden. Dabei sollen sie in der kompakteren Stencil-Form dargestellt werden.

$$\frac{1}{4}\begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 3 & 2 & 0 \\ 0 & 2 & 3 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.6}$$

$$\frac{1}{4}\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \tag{3.7}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix} \tag{3.8}$$

3.6 ist die Prolongation nach Wesseling [42, S.64], 3.7 die Prolongation nach Kwak [23] und 3.8 die bilineare Interpolation z.B. in [42, S.68]. Im Gegensatz

zu einfachen Interpolation oder Injektion verwenden diese Operatoren auch die Informationen über die Nachbarn. Sie sind u.a. interessant, weil allgemein die Gleichung

$$m_R + m_P > 2m$$

[42, S.252] gelten soll. m_P und m_R stehen dabei für die jeweilige Ordnung der Operators und $2m$ für die Ordnung des Gleichungssystems. Da es sich im Fall der Navier-Stokes-Gleichungen um $2m = 2$ handelt, benötigt man mindestens einen Operator 2.Ordnung.
Ein Stencil wie folgt zu interpretieren. Sei als Beispiel folgendes Netz gegeben.

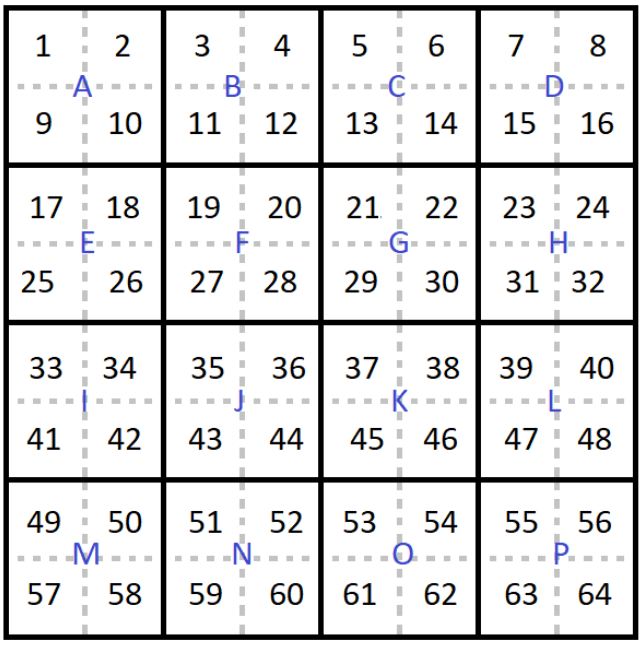


Abbildung 3.8: Beispiel eines Netzes für einen Stencil

Der Prolongationsoperator nach Kwak 3.7 hat für die feine Zelle 20 für eine Größe U die Form

$$U_{20} = \frac{1}{4}(2U_F + U_B + U_G)$$

Es gilt für den Stencil von Prolongation und Restriktion die Beziehung

$$T_R = \sigma T_P^\star$$

[42], S.70. Der Operator $P^\star$ wird durch Transponieren der Matrix P erzeugt und da es sich in allen vorgestellten Stencil um symmetrisch Matrizen handelt, gilt also $P = P^\star$. Der Skalierungsfaktor σ ergibt sich als $\frac{1}{4}$ im zweidimensionalen Fall, da nach [42], S.71 σ als

$$(\frac{h_{fein}}{h_{grob}})^d$$

mit d die Dimension des Netzes, h_{fein} die Schrittweite im feinen Netz und h_{grob} die Schrittweite des groben Netzes ist. Da in den die Agglomeration eine doppelte Schrittweite erzeugt und im zweidimensionalen vorliegt, ergibt sich der oben genannte Faktor.
Somit erhält man als Restriktionsoperator für die grobe Zelle F

$$U_F = \frac{1}{16}(2U_{19}+2U_{20}+2U_{27}+2U_{28}+U_{11}+U_{12}+U_{18}+U_{21}+U_{26}+U_{29}+U_{35}+U_{36})$$

Eine Übersicht und genauere Untersuchung dieser drei Operatoren und die Anpassungen für verschiedene Randbedingungen findet sich in [30].

3.5 Glätter

Der Glätter des Mehrgitterverfahrens wird passend zum Problem ausgewählt. Im Fall des FAS wird ein Verfahren zum Lösen nichtlinearer Gleichungen gewählt, während bei den klassischen Mehrgitterverfahren beispielsweise mehrmals ein Jacobi- oder ein Gauß-Seidel-Verfahren angewendet wird.
Zum Glätten der Navier-Stokes oder Euler-Gleichung hat sich die Verwendung von Verfahren auf Grundlage der Runge-Kutta-Verfahren durchgesetzt. So wurden sie beispielsweise von Jameson et al. zur Lösung der Euler-Gleichungen verwendet [17]. Dabei haben Verfahren, welche an implizite Runge-Kutta-Verfahren angelehnt sind, die bisher besten Ergebnisse geliefert, wie z.B. in [39].

In diesem Fall soll das s-stufige diagonal implizite Runge-Kutta-Verfahren mit einem Newton-Schritt zum Linearisieren des Problems verwendet werden [24, S.77ff].
Das Runge-Kutta-Verfahren gehört zu den Einschrittverfahren und hat allgemein die Form

$$y_n = y_{n-1} + h_n \sum_{r=1}^{s} b_r k_r$$

mit $k_r = f(t + h_n c_r, x + h_n \sum_{i=1}^{s} a_{ri} k_i)$ zu einer gegebenen Differentialgleichung $y' = f(t, x)$ [34].
Das Runge-Kutta-Verfahren wird meist mithilfe eines Butcher-Schemas angegeben, welches die relevanten Koeffizienten enthält.

Tabelle 3.4: Butcher-Schema

c	**A**
	b^T

mit

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{1s} \\ \vdots & \ddots & \vdots \\ a_{s1} & \cdots & a_{ss} \end{pmatrix}, \qquad b = \begin{pmatrix} b_1 \\ \vdots \\ b_s \end{pmatrix}, \qquad c = \begin{pmatrix} c_1 \\ \vdots \\ c_s \end{pmatrix}$$

In Fall des diagonal-impliziten-Runge-Kutta-Verfahrens hat man folgende Einträge des Butcher-Schemas.

$$\mathbf{A} = \begin{pmatrix} \alpha_{11} & 0 & \cdots & 0 \\ \alpha_{21} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & \alpha_{s,s-1} & \alpha_{ss} \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \alpha_{s+1,s} \end{pmatrix}, \qquad c = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

Somit ergibt sich durch Anwendung des oben beschriebenen Runge-Kutta-

Verfahrens auf die in 2.19 gegebene Form folgende Gleichungen mit $h_n = \Delta t$

$$\begin{aligned} k_1 &= -M^{-1}R(W^{T_n} + \alpha_{11}\Delta t k_1) \\ k_2 &= -M^{-1}R(W^{T_n} + \alpha_{21}\Delta t k_1 + \alpha_{22}\Delta t k_2) \\ &\vdots \\ k_s &= -M^{-1}R(W^{T_n} + \alpha_{s,s-1}\Delta t k_{s-1} + \alpha_{ss}\Delta t k_s) \\ W^{T_{n+1}} &= W^{T_n} + \alpha_{s+1,s}\Delta t k_s \end{aligned}$$

Um die Lösung des nichtlinearen Systems zu erhalten, verwendet man die Nullstellenfunktion

$$g_j(k_j) = k_j + M^{-1}R(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1}) + \alpha_{jj}\Delta t k_j)$$

für die man ein Schritt des Newton-Verfahrens

$$k_j^{(m+1)} = k_j^{(m)} - g_j(k_j^{(m)})(\frac{\partial g_j(k_j^{(m)})}{\partial k_j})^{-1} \tag{3.9}$$

ausführt. Die dazu benötigte Ableitung hat die Form

$$\frac{\partial g_j(k_j^{(m)})}{\partial k_j} = I + \alpha_{jj}\Delta t M^{-1}\frac{\partial R}{\partial W}(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1} + \alpha_{jj}\Delta t k_j^{(m)})$$

Als Startwert $k_j^{(0)}$ wird Null angenommen.
Somit erhält man

$$\begin{aligned} k_j = k_j^{(1)} &= k_j^{(0)} - g_j(k_j^{(0)})[\frac{\partial g_j(k_j^0)}{\partial k_j}]^{-1} \\ &= -g_j(k_j^{(0)})[I + \alpha_{jj}\Delta t M^{-1}\frac{\partial R}{\partial W}(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1} + \alpha_{jj}\Delta t k_j^0)]^{-1} \\ &= -[P_j]^{-1}\left(g_j(k_j^{(0)})\right) \end{aligned}$$

mit

$$P_j = I + \alpha_{jj}\Delta t M^{-1}\frac{\partial R}{\partial W}(W^{T_n} + \alpha_{j,j-1}\Delta t k_{j-1})$$

Wendet man dies zusammen auf das implizite Runge-Kutta-Verfahren an, so erhält man folgenden Algorithmus zur Berechnung der k_j.

$$
\begin{aligned}
k_1 &= -[P_1]^{-1} M^{-1} R(W^{T_n}) \\
k_2 &= -[P_2]^{-1} M^{-1} R(W^{T_n} + \alpha_{21} \Delta t k_1) \\
&\vdots \\
k_s &= -[P_s]^{-1} M^{-1} R(W^{T_n} + \alpha_{s,s-1} \Delta t k_{s-1}) \\
W^{T_{n+1}} &= W^{T_n} + \alpha_{s+1,s} \Delta t k_s
\end{aligned}
$$

Seien nun

$$
\begin{aligned}
W^{(0)} &= W^{T_n} \\
W^{(j)} &= W^{T_n} - \alpha_{j+1,j} \Delta t [P_j]^{-1} M^{-1} R(W^{j-1})
\end{aligned}
$$

Daraus folgt mittels Induktion, dass man das Runge-Kutta-Verfahren wie folgt umschreiben kann

$$
\begin{aligned}
W^{(0)} &= W^{T_n} \\
W^{(1)} &= W^{(0)} - \alpha_{21} \Delta t [P_1]^{-1} M^{-1} R(W^{(0)}) \\
&\vdots \\
W^{(s)} &= W^{(0)} - \alpha_{s+1,s} \Delta t [P_s]^{-1} M^{-1} R(W^{(s-1)}) \\
W^{T_{n+1}} &= W^{(s)}
\end{aligned}
$$

Wie man sieht, muss daher für jeden Schritt die lineare Gleichung

$$
P_j h_j = \alpha_{j+1,j} \Delta t M^{-1} R(W^{(j-1)})
$$

lösen um dadurch auf h_j zu gelangen. Durch Einsetzen von P_j ergibt sich somit

$$
\begin{gathered}
\left(I + \alpha_{jj} \Delta t M^{-1} \frac{\partial R}{\partial W}(W^{(j-1)}) \right) h_j = \alpha_{j+1,j} \Delta t M^{-1} R(W^{(j-1)}) \\
\Updownarrow \\
\left((\Delta t)^{-1} M + \alpha_{jj} \frac{\partial R}{\partial W}(W^{(j-1)}) \right) h_j = \alpha_{j+1,j} R(W^{(j-1)})
\end{gathered}
$$

.

Im nächsten Schritt wird die Ableitung $\frac{\partial R}{\partial W}$ berechnet. Eine Möglichkeit wäre

es die Jacobi-Matrix zu verwenden. Diese würde aber für die Residuen, welche über die Variablen an den Kanten berechnen werden, zu aufwändig, da man hierfür nicht nur die Nachbarn, sondern auch die Nachbarn der Nachbarn benötigt. Da die Ableitung jedoch lediglich im Glätter verwendet wird, reicht es eine Approximation zu verwenden. Diese soll nur mithilfe der Zelle und seiner Nachbarn berechnet werden.
Für den konvektiven Teil wählt man die folgende Näherung

$$\oint_{\partial\Omega_i} \overrightarrow{F}_C \,\mathrm{d}S \approx \sum_{j\in\mathcal{N}(i)} \frac{1}{2}\left((\overrightarrow{F}_C^{ij})(W_i) + (\overrightarrow{F}_C^{ij})(W_j)\right) - \frac{1}{2}|B_{ij}^{Roe}|\,(W_j - W_i)$$

Sind die B_{ij}^{Roe} lokal konstant, so kann man mit der Ableitung des konvektiven Teils die Ableitung $\frac{\partial R_i}{\partial W_i}$ wie folgt approximieren

$$\frac{\partial(\oint_{\partial\Omega_i} \overrightarrow{F}_C \,\mathrm{d}S)}{\partial W_i} \approx \widehat{\frac{\partial R_i}{\partial W_i}} = \frac{1}{2}\begin{cases} \sum_{j\in\mathbb{N}(i)} |A_{ij}^{Roe}| & k = i \\ -|A_{jk}^{Roe}| & k \in \mathbb{N}(i) \\ 0 & k \neq i, k \notin \mathbb{N}(i) \end{cases}$$

Für den viskosen Teil kann man eine Approximation mittels Green-Gauss wie in 2.4 verwenden.
Eine weitere Verbesserung kann noch dadurch erzielt werden, dass man sich in erster Linie nur für den stationären Zustand interessiert. Daher kann man Δt mit $\Delta T = diag(diag(\Delta t_i))$ wobei der lokale Zeitschritt Δt_i wie folgt definiert ist.

$$\begin{aligned}\Delta t_i = CFL * vol(\Omega_i)[&\sum_{j\in\mathbb{N}(i)} \frac{1}{2}(|V_{ij}| + a_{ij}A_{ij}) \\ &+ \frac{8(\mu_{eff})_{ij}A_{ij}}{||x_{p_i} - x_{p_j}||_2\rho_i}\left(max\{\frac{4}{3}, \frac{(\kappa_{eff})_{ij}(\gamma-1)}{(\mu_{eff})_{ij}}\}\right)]^{-1}\end{aligned}$$

Diese Definition stammt aus einer Näherung des Spektralradiuses der diagonalen Blöcke von $\frac{\partial R}{\partial W}$ [siehe [25], S.165).
Zudem wird noch ein Relaxationsparameter ϵ verwendet, sodass man zu

$$\left((\Delta T)^{-1}M + \epsilon\alpha_{jj}\widehat{\frac{\partial R}{\partial W}}\right) h_j = \alpha_{j+1,j}R(W^{(j-1)}) \tag{3.10}$$

bzw.

$$P_j h_j = \alpha_{j+1,j} R(W^{(j-1)}) \tag{3.11}$$

mit $P_j = (\Delta T)^{-1} M + \epsilon \alpha_{jj} \frac{\widehat{\partial R}}{\partial W}$ kommt.
In jedem Schritt des Runge-Kutta-Verfahrens muss ein lineares Gleichungssystem wie in 3.11 definiert gelöst werden. Dazu verwendet man ein symmetrisches Block Linien Gauss-Seidel-Verfahren. Es gelte

$$P = P_j, \quad h = h_j, \quad b = \alpha_{j+1,j} R(W^{(j-1)})$$

Für $i = 1, \ldots, n$ sei gegeben.

$$G_i \subset G \qquad G_j \cap G_i = \emptyset, \qquad i \neq j, \qquad \bigcup_{j=1}^{n} G_j = G, \qquad r_i = \sharp(G_i)$$

Ist $r_i > 1$, so nennt man $G_i = \{l_i^1, \ldots, l_i^{r_i}\}$ eine Linie. Sonst ist G_i ein Punkt. Die Linien aggregieren dabei Elemente, die eine Anisotropie repräsentieren. Eine genauere Erklärung findet sich beispielsweise in [25, S.148ff.]
Entlang einer Linie ist die dazugehörige Block-Tridiagonalmatrix durch

$$\mathrm{Tri}_{G_i} = (\Delta t_{G_i})^{-1} M_{G_i} + \alpha_{jj} \begin{pmatrix} \frac{\partial R_{l_i^1}}{\partial W_{l_i^1}} & \frac{\partial R_{l_i^1}}{\partial W_{l_i^2}} & 0 & \cdots & 0 \\ \frac{\partial R_{l_i^2}}{\partial W_{l_i^1}} & \frac{\partial R_{l_i^2}}{\partial W_{l_i^2}} & \frac{\partial R_{l_i^2}}{\partial W_{l_i^3}} & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \frac{\partial R_{l_i^{r_i-1}}}{\partial W_{l_i^{r_i-2}}} & \frac{\partial R_{l_i^{r_i-1}}}{\partial W_{l_i^{r_i-1}}} & \frac{\partial R_{l_i^{r_i-1}}}{\partial W_{l_i^{r_i}}} \\ 0 & \cdots & 0 & \frac{\partial R_{l_i^{r_i}}}{\partial W_{l_i^{r_i-1}}} & \frac{\partial R_{l_i^{r_i}}}{\partial W_{l_i^{r_i}}} \end{pmatrix}$$

$$h_{G_i}^{m+1} = \mathrm{Tri}_{G_i}^{-1} \left(b_{G_i} - \sum_{j \in G_1, \ldots G_{i-1}, j \notin G_i} P_{G_i,j}\, h_j^{k+1} - \sum_{j \in \{1,\cdots,n\} \setminus \{G_1, \ldots G_i\}} P_{G_i,j}\, h_j^{k} \right)$$

Davon führt man 5 symmetrische sweeps durch. Dazu wird ein sweep über alle Linien aus und anschließend über alle Punkte, die auf keiner Linie liegen, ausgeführt. Im Anschluss führt man das Ganze umgekehrt aus, d.h. man geht zuerst über alle Punkte, die nicht auf einer Linie liegen und dann über alle Linien.

Kapitel 4

Implementierung

In diesem Kapitel soll auf die neuen Komponenten und deren Implementierung eingegangen werden.

4.1 Agglomeration der Grobnetze

Zum Agglomerieren der Grobnetze wurde einerseits ein neuer, an den strukturierten Netzen angelehnter Algorithmus und als Vergleichsalgorithmus der bereits implementierte MGridGen-Algorithmus verwendet.
Um den Algorithmus zu realisieren, werden die Netzinformationen benötigt. Diese enthalten folgende Eigenschaften.

- Gesamtanzahl der Netzzellen
- x-, y- und z- Koordinate des Mittelpunktes jeder Netzzelle
- die zu einer Zelle zugehörigen Kanten mit ihrer Normalen
- Volumen der Netzzelle
- Information, ob es sich um eine Zelle am Rand handelt
- Gesamtanzahl der Kanten
- zu einer Kante zugehörige Netzzellen

Sowohl der MGridGen-Algorithmus als auch der neue Algorithmus benötigen zum Erzeugen der Agglomeration Informationen über die Nachbarn. MGridGen arbeitet mit dem sogenannten CSR Format(„compressed storage format“, komprimiertes Speicherformat). Dieses wird oft zur Speicherung schwachbesetzter Graphen verwendet und hat den Vorteil, dass man bei n Netzelementen und m Kanten nicht Matrizen mit der Größe $n \times n$ zur Speicherung benötigt, sondern mithilfe zwei Vektoren der Größe $n + 1$ (hier Neigh_id) und $2m$ (hier Neighbors). Für eine Zelle i enthalten Neigh_id[i] und Neigh_id[i+1]-1 die Indices aus dem Vektor Neighbors, zwischen denen die Nachbarn gespeichert sind. [31] Für ein Testnetz würde folgende Struktur erzeugt werden.

Abbildung 4.1: Testnetz

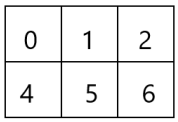


Tabelle 4.1: Nachbarinformationen im CSR Format

Neigh_id	0	2	5	7	9	12	14							
Neighbors	1	4	0	2	5	1	6	0	5	1	4	6	2	5

Um diese Speicherung an den vorliegenden Daten zu erzeugen, wird folgender Algorithmus verwendet.
Der Algorithmus initialisiert zunächst die Anfangsdaten. Dann geht er schrittweise alle Zellen $j \in \{0, \cdots, Anz\_Punkte - 1\}$ des Netzes durch. Für jede Kante k der Zelle j wird nun überprüft, welche der beiden zur Kannte zugehörigen Zellen nicht der Zelle j entspricht. Diese andere Zelle ist dann der Nachbar von j und wird dementsprechend in den Vektor Neighbors aufgenommen. Zudem wird die Zählvariable i jedem neuen Nachbarn um eins erhöht. Nachdem alle Kanten der Zelle j durchlaufen wurden, wird der Wert für die nächste Zelle $j + 1$ in Neigh_id auf i gesetzt.

Algorithm 1 Nachbarnsuche

```
1: procedure NACHBARN(Netzstruktur)   ▷ Sucht alle Nachbarn anhand der Kanteninformationen
2:     i = 0
3:     Neigh_id[0] = 0
4:     for jede Zelle j do
5:         for jede Kante k der Zelle j do
6:             a ← 1.Zelle der Kante k
7:             b ← 2.Zelle der Kante k
8:             if j=b thenNeighbors[i] = a
9:             else
10:                Neighbors[i] = b
11:            end if
12:            i = i + 1
13:        end for
14:        Neigh_id[j + 1] = i
15:    end for
16:    return Neigh_id, Neighbors
17: end procedure
```

Bei Mgridgen handelt es sich um einen Agglomerationsalgorithmus, der anhand folgender Informationen der Elemente eines feinen Netzes ein gröberes generiert.

- Anzahl der Elemente
- Volumen der Elemente
- Nachbarn der Elemente
- Oberfläche der Seitenflächen der Elemente

Der Algorithmus versucht möglichst wohlgeformte neue Elemente unter Einbeziehung einer Minimal- und Maximalanzahl an Unterzellen zu generieren. Für jedes der neuen Elemente wird dafür die Kennzahl $S = \frac{l^2}{A}$ im zweidimensionalen Fall bzw. $S = \frac{A^{1,5}}{V}$ im dreidimensionalen Fall berechnet. A steht für die Oberfläche des Elements, V für das Volumen und l für den Umfang. Dieses gibt an wie sehr das neue Element einer Kugel (im dreidimensionalen) bzw. einem Kreis (im zweidimensionalen) ähnelt wobei ein kleines S für ein

gutes Verhältnis steht. Es gibt vier Optionen wie die Güte des neuen Netzes M^{k+1} gemessen wird. Dabei sucht man immer das Minimum der folgenden Varianten über alle möglichen neuen Netze.

1. Summe der Kennzahl S aller neuen Elemente
2. mit der Anzahl der Unterzellen des neuen Elements gewichtete Summe der Kennzahl S über alle neuen Elemente
3. Schlechteste Kennzahl S aller neuen Elemente
4. Kombination des 2. und 3. Punktes. Zunächst sucht man alle Netze mit der kleinsten schlechtesten Kennzahl und sucht dann nach dem kleinsten 2. Kriterium, der gewichteten Summe

Um das neue Verfahren anwenden zu können, müssen die Netze einige Voraussetzungen erfüllen.

- Anzahl der Elemente im Netz durch 4^k (mit k Anzahl der Vergröberungen) teilbar
- Netzstruktur liegt in 2D vor
- viereckige Elemente
- farfield besitzt maximale x-Koordinate

Um das neue vergröberte Netz zu bestimmen, wird dann folgende Vorgehensweise verwendet

1. Es wird eine Liste der Elemente mit maximaler x-Koordinate erstellt.
2. In der Liste wird aus allen Elementen, die noch keine zugeordnete Grobzelle besitzen, das Start- (minimale z-Koordinate) und Stoppelement (maximale z-Koordinate) bestimmt.
3. Dem Startelement und allen Nachbarelementen, die noch keine zugeordnete Grobzelle besitzen, wird ein Grobzelle zugeordnet (siehe Abbildung 4.3)

4. Die Nachbarelemente, denen im letzten Schritt eine Grobzelle zugeordnet wurden, werden wiederum auf ihre Nachbarn überprüft. Dabei wird der Zelle, die Nachbar aller untersuchten Nachbarn ist, die verbliebene vierte Zelle im der neuen Grobzelle und wird dieser dementsprechend zugeordnet. (siehe Abbildung 4.1)

5. Diejenigen Elemente, die Nachbarn der Nachbarn sind, aber nicht die vierte Zelle sind, sind die möglichen Kandidaten für den nächsten Start. Dabei wird dasjenige Element gewählt, welches selbst genau zwei Nachbarn besitzt, die noch keiner Grobzelle zugeordnet wurden. Wenn das Startelement noch das aus Schritt 2 ist, so trifft diese Bedingung auf zwei Elemente zu. Deshalb wird in diesem Fall das Element mit der kleineren x-Koordinate als neues Startelement gewählt. (siehe Abbildung 4.5)

6. Dann werden die Schritte 3-5 wiederholt, bis dem Stoppelement eine Grobzelle zugeordnet wurde.

7. Es wurden nun zwei Schichten bezüglich der z-Richtung bearbeitet. Um die verbleibenden Schichten zu bearbeiten werden die Schritte 2-6 wiederholt, bis allen Elementen eine neue Grobzelle zugeordnet wurde.

   Um eine Sequenz an Vergröberungen zu erzeugen, wird, wie bei MGridGen, das Verfahren auf das vergröberte Netz erneut ausgeführt.

An einem Beispielnetz sieht das dann in grafischer Form wie folgt aus.

Abbildung 4.2: Schritt 1 und 2: Netz mit Liste der Randelemente gegeben durch die Zahlen $1, 2, \ldots,$ wobei 1 das Startelement ist

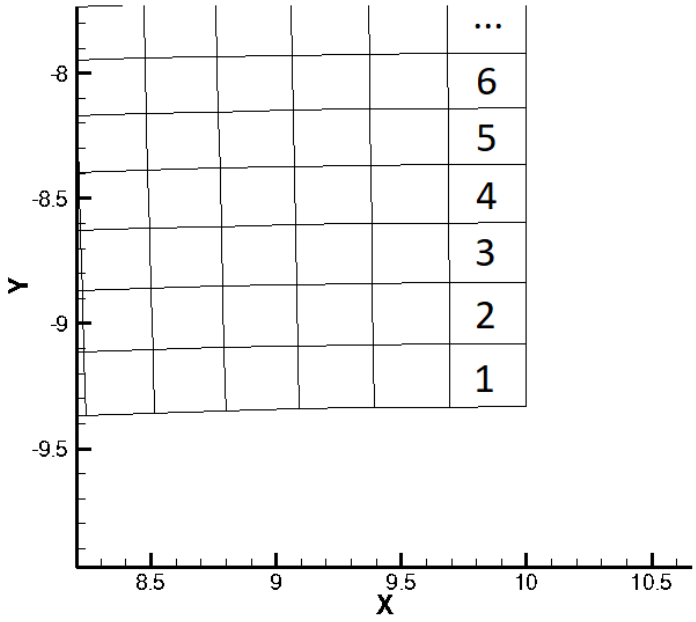
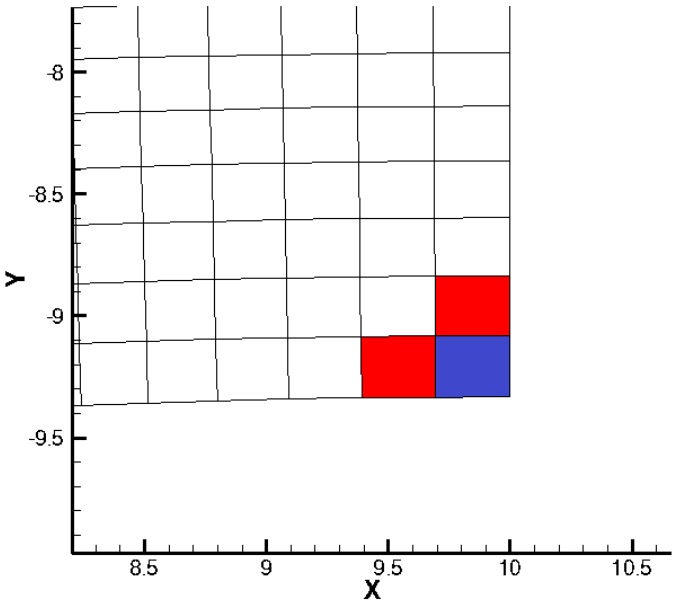


Abbildung 4.3: Schritt 3: Die blaue Zelle ist die Startzelle und die roten Zellen kennzeichnen die Nachbarzellen der blauen Zelle. Diesen 3 Zellen wird die Grobzelle 1 zugeordnet



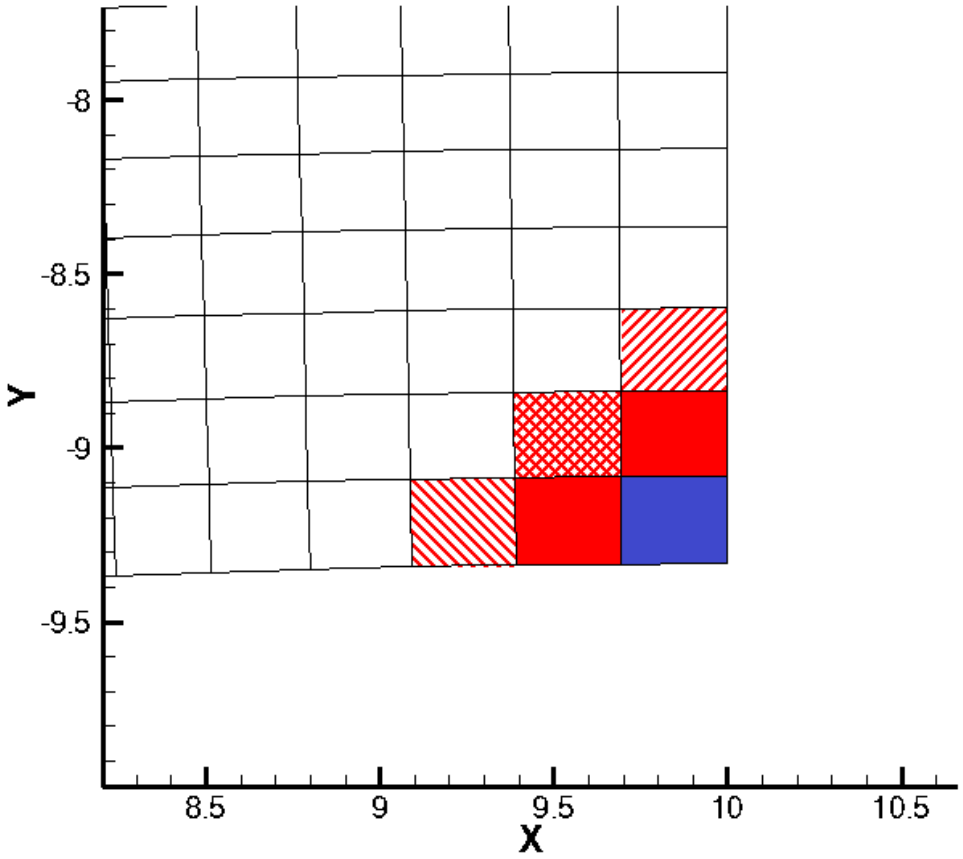


Abbildung 4.4: Schritt 4: Die rot schraffierten Zellen sind die Nachbarn der roten Zellen. Ein weiterer Nachbar ist die blaue Zelle, aber da diese bereits die Grobzelle 1 zugeordnet bekommen hat, wird sie ignoriert. Die Schraffur von oben links nach unten rechts markiert die Nachbarzellen der roten Zelle links neben der blauen Zelle und die Schraffur von oben rechts nach unten links bezeichnet die Nachbarn zur roten Zelle oberhalb der blauen Zelle. Die Zelle, die beide Schraffuren besitzt, ist die gesucht 4. Zelle der neuen groben Zelle und wird dementsprechend auch die Grobzelle 1 zugeordnet.

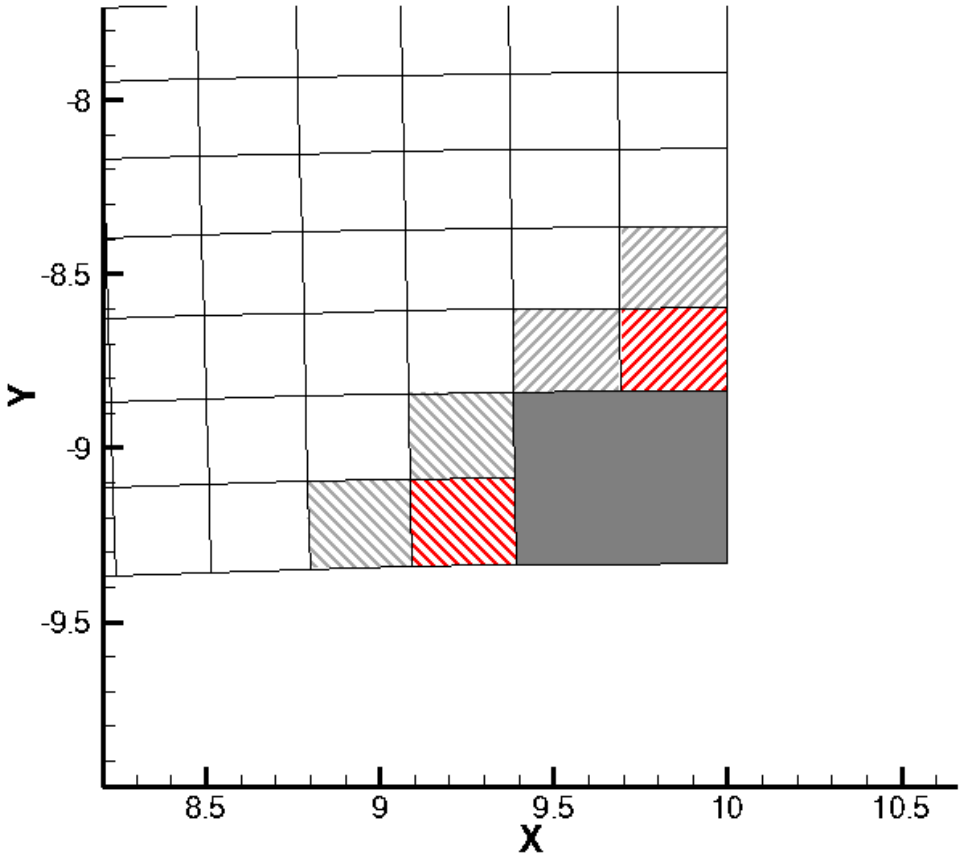


Abbildung 4.5: Schritt 5: Die große, graue Zelle ist die neue Grobzelle 1. Die rot schraffierten Zellen sind die noch übrigen Nachbarn aus Schritt 4. Für diese sucht man nun die Nachbarn, die noch nicht einer Grobzelle zugeordnet wurden und zählt deren Anzahl. Beide besitzen 2 Nachbarn, die dieses Kriterium erfüllen. Diesen Sonderfall, wenn man mit einem Element aus der Liste der maximalen y-Koordinate gestartet ist, behandelt man, indem man die rot schraffierte Zelle wählt, die eine kleinere x-Koordinate besitzt, also in diesem Fall die linke rot schraffierte Zelle. Diese ist nun die neue Startzelle.

Abbildung 4.6: Schritt 3: Nun wiederholt man den dritten Schritt mit der neuen Startzelle. Die blaue Zelle ist wieder die Startzelle und die roten Zellen kennzeichnen deren Nachbarzellen. Diesen drei Zellen wird die Grobzelle 2 zugeordnet. Der andere Nachbar von der blauen Zelle hat bereits zuvor eine neue Grobzelle zugeordnet bekommen und wird daher ignoriert.

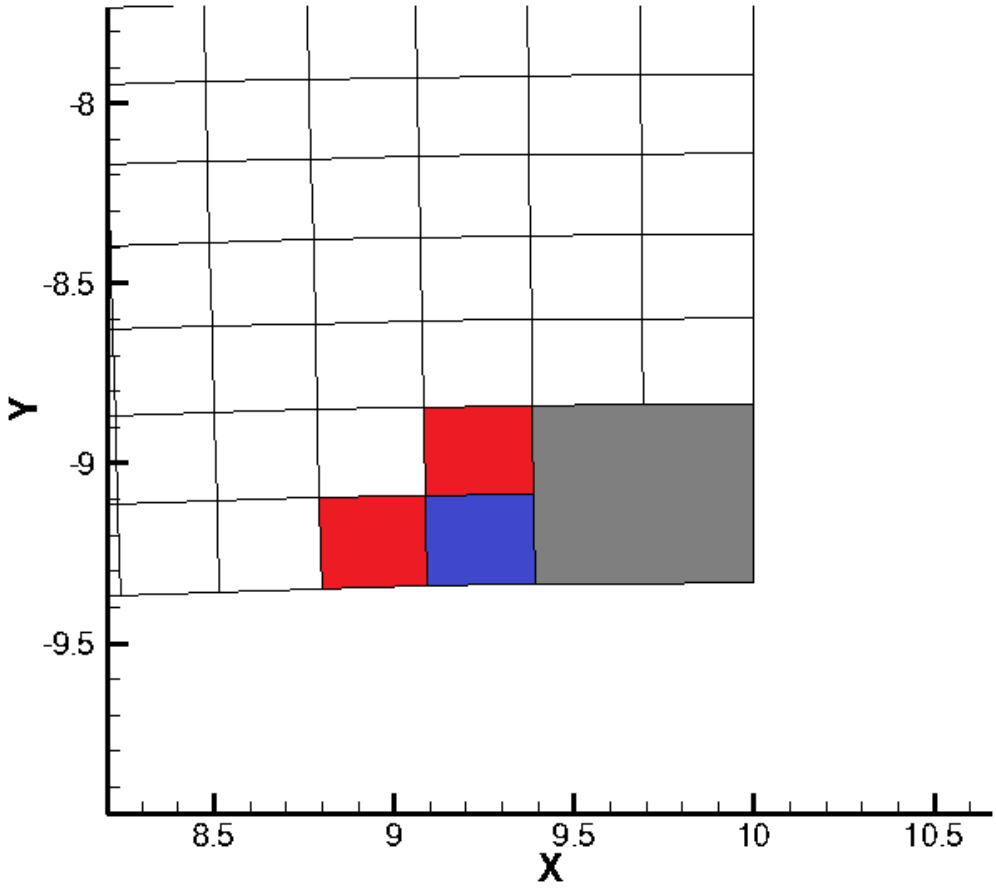


Abbildung 4.7: Schritt 4: Die rot schraffierten Zellen bezeichnen wieder die Nachbarn der roten Zellen. Weitere Nachbarn sind die blaue Zelle und eine Unterzelle der grauen Zelle, aber da diese bereits eine Grobzelle zugeordnet bekommen haben, werden sie ignoriert. Die Schraffur von oben links nach unten rechts markiert wieder die Nachbarzellen der roten Zelle links neben der blauen Zelle und die Schraffur von oben rechts nach unten links bezeichnet die Nachbarn zur roten Zelle oberhalb der blauen Zelle. Die Zelle, die beide Schraffuren besitzt, ist die gesuchte Zelle und wird der Grobzelle 2 zugeordnet.

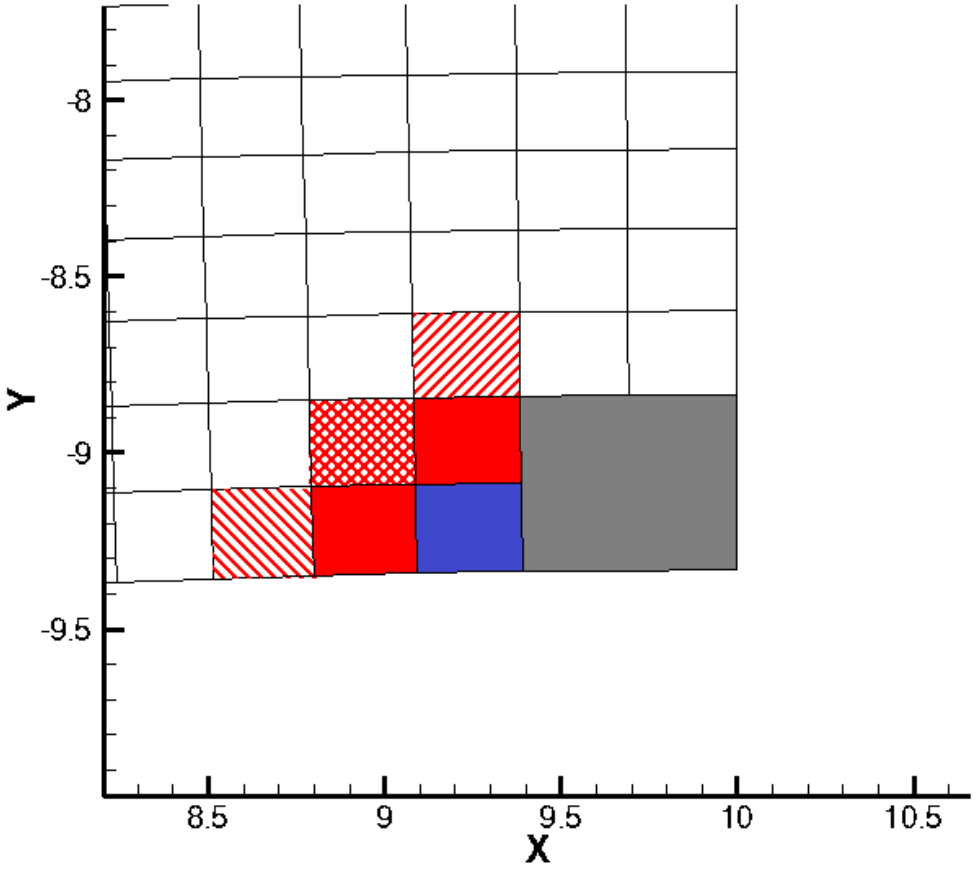
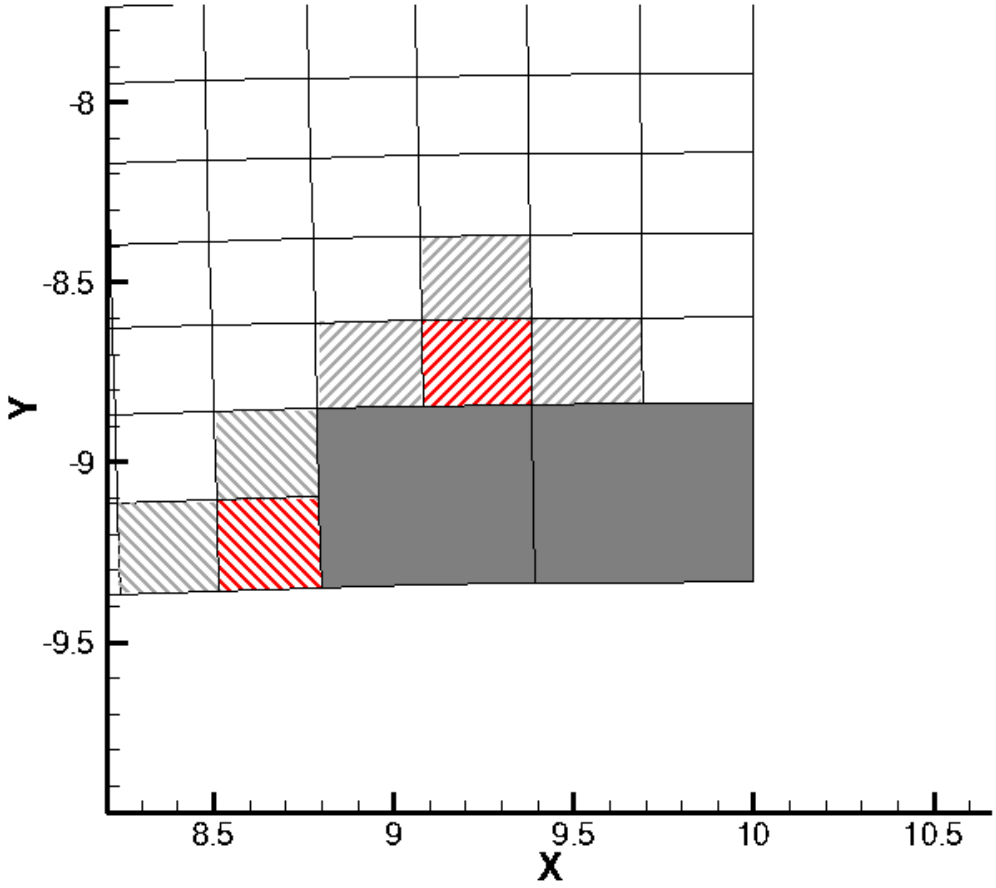


Abbildung 4.8: Schritt 5: Die großen, grauen Zelle bezeichnen die groben Zellen 1 und 2. Die rot schraffierten Zellen bezeichnen wieder die übrigen Nachbarn aus dem vorherigen Schritt. Deren noch nicht zugeordneten Nachbarn erhalten wieder die passende Schraffur. Die obere rote Zelle besitzt 3 noch nicht zugeordnete Nachbarn, während die auf dem Bild linke rot schraffierte Zelle nur 2 besitzt. Somit ist letztere die neue Startzelle.



Auch am inneren Rand funktioniert der Algorithmus genauso.

Abbildung 4.9: Schritt 3: Die grauen Zelle seien wieder die bereits einer Grobzelle zugeordneten Zellen. Dann werden im 3.Schritt wieder der Startzelle und deren Nachbarn eine neue Grobzelle j zugeordnet

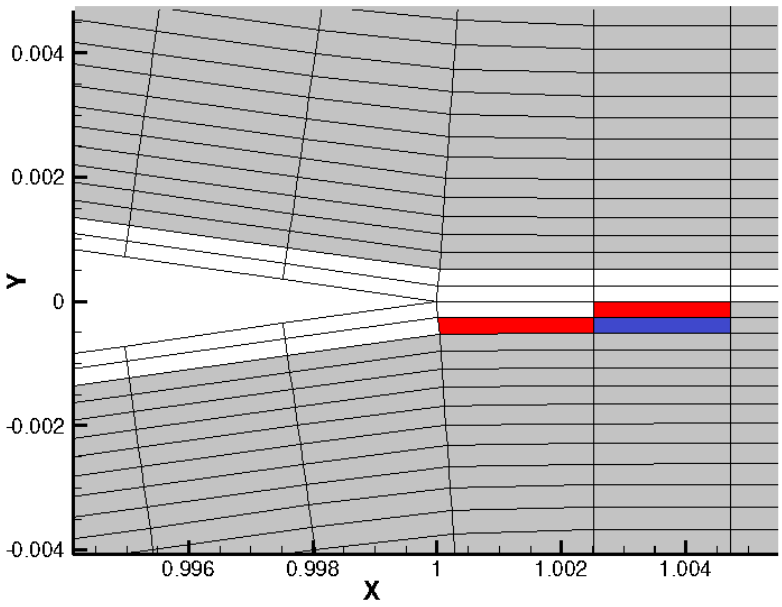


Abbildung 4.10: Schritt 4: Man kennzeichnet die nach der roten Zellen durch eine rote Schraffur.

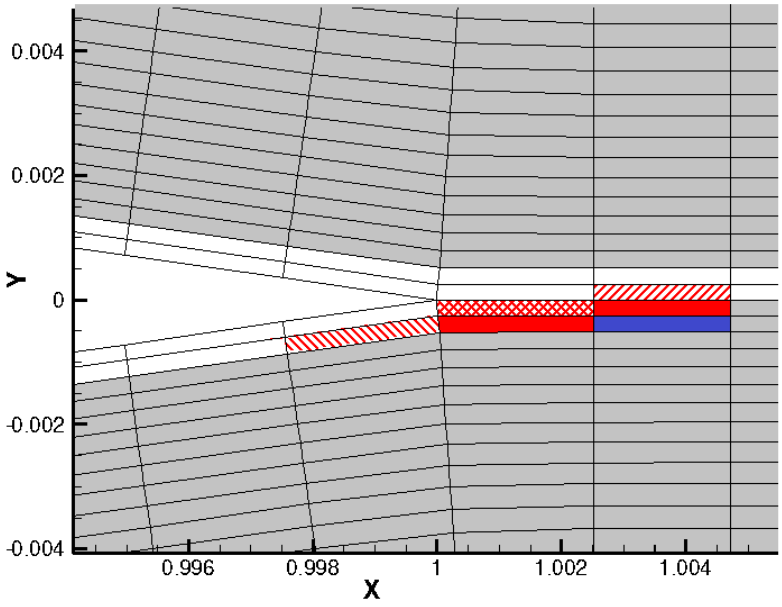
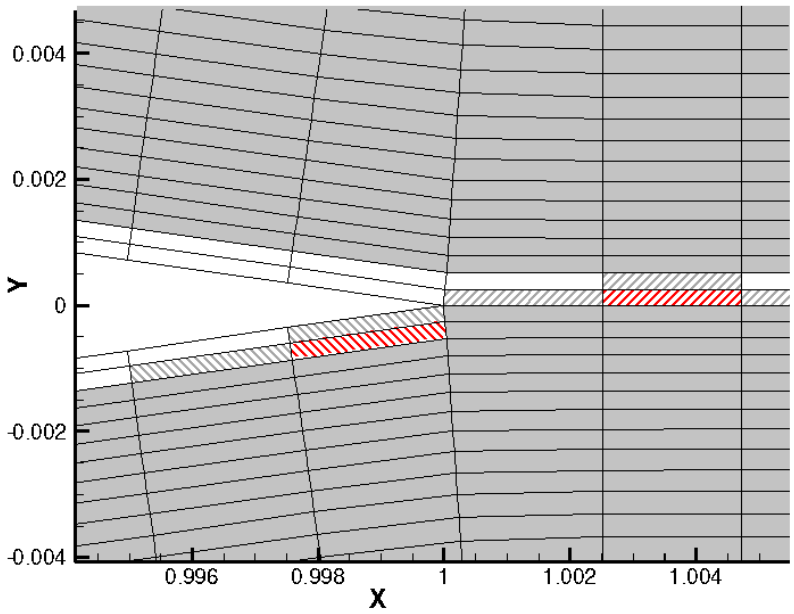


Abbildung 4.11: Schritt 5: Es wird wieder die Anzahl der Nachbarn der rot schraffierten Zellen gesucht. Die mit 2 Nachbarn ist die neue Startzelle



Ein Netz hat schematisch folgende Darstellung.

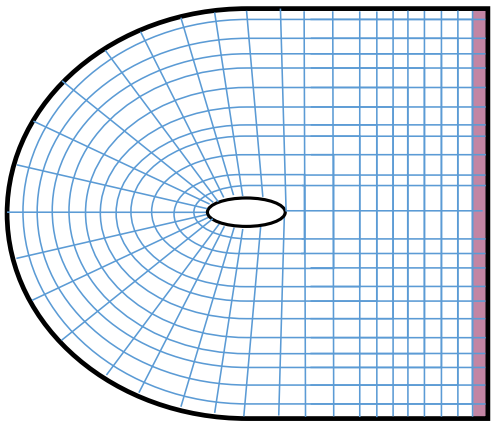


Abbildung 4.12: Schema eines Netzes

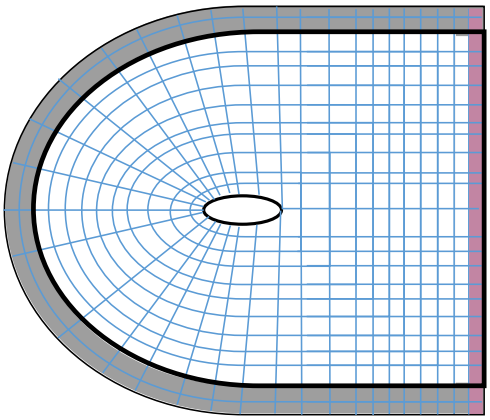


Abbildung 4.13: Schema eines Netzes nach dem 6.Schritt.

Die lilafarbenen Zellen stehen für die Elemente aus list_max_x. Die Abbildung 4.1 zeigt das Netz nachdem zwei Schichten bearbeitet wurden. Die grauen Zellen bezeichnen die bereits zugeordneten Zellen. Man sieht, dass die Zellen innerhalb der breiten schwarzen Linie wieder ein Schema wie in 4.1 bilden.

Als Pseudocode hat der Code die folgende Form.

Algorithm 2 Grobnetzagglomeration

```
1: procedure NETZ_GROB(Netzstruktur, Neigh_id, Neighbors)  ▷
   Agglomeriert ein den strukturierten Netzen ähnliches Grobnetz mittels
   der Netz- und Nachbarinformationen und speichert dies in part
2:     Finde unter allen Zellen i die maximale x-Koordinate max_x
3:     for jede Zelle i do
4:         if z-Koordinate von i=max_z then i in list_max_z aufnehmen
5:         end if
6:         part[i] negativen Wert zuordnen
7:     end for
8:     Finde die Zelle i in list_max_z mit min. z-Koord. und setze start=i
9:     Finde die Zelle i in list_max_z mit max. z-Koord. und setze stop=i
10:    next=start
11:    for jede Zelle i am Rand do
12: findet alle Elemente am Rand auf der letzten Seite (nötig, da es bei
   den höhren Agglomerationen zu Verschiebungen kommt und die neuen
   Elemente nicht mehr exakt auf einer Linie liegen)
13:        if next=stop then for-Schleife verlassen
14:        end if
15:        if Existiert ein Nachbar i von next in list_max_z then next=i
16:        elsenext=Nachbar von next mit maximaler z-Koordinate
17:        end if
18:    end for
19:    it=0
20:    for jede Zelle i do
21:        part[start]=it
```

```
22:         for jede Nachbarzelle j der Zelle start do
23:             if part[j]<0 then
24:                 part[j]=it
25:                 j in list_neigh aufnehmen
26:             end if
27:         end for
28:         for jede Zelle j in list_neigh do
29:             jede Nachbarzelle k der Zelle j mit part[k]<0 in list_neigh_2
    aufnehmen
30:         end for
31:         doppelt auftretende Zelle k in list_neigh_2 finden und part[k]=it
    setzen
32:         for alle Zellen l in list_neigh_2 außer k do
33:             if start ist aus list_max_x then Zelle mit minimaler x-
    Koordinate finden und next=l
34:             else Zelle mit genau 2 Nachbarn m mit part[m]<0 finden und
    next=l setzen
35:             end if
36:         end for
37:         if part[stop]>=0 then //neue Schicht Finde die Zelle l in
    list_max_z mit minimaler z-Koordinate und part[l]<0 und setze start=l;
    Finde die Zelle l in list_max_z mit maximaler z-Koordinate und
    part[l]<0 und setze stop=l
38:         elsestart=next
39:         end if
40:         if part[start]>=0 then For-Schleife beenden
41:         end if
42:         it=it+1
43:     end for
44:     return part
45: end procedure
```

4.2 Operatoren

Die Prolongations- und Restriktionsoperatoren werden ähnlich wie Agglomeration generiert.
Für die Prolongation sei ein Stencil mit der Form

$$\frac{1}{f_P}\begin{bmatrix} v_5 & v_3 & v_4 & v_6 \\ v_3 & v_1 & v_2 & v_4 \\ v_4 & v_2 & v_1 & v_3 \\ v_6 & v_4 & v_3 & v_5 \end{bmatrix}$$

vorgegeben. Seien S die gesuchten Werte des feinen Netzes und T die Werte des groben Netzes. Außerdem sei durch $coarse[i]$ die zur Zelle i des feinen Netzes die dazugehörige agglomerierte Zelle des groben Netzes gegeben.

1. Es wird eine Liste der Elemente des feinen Netzes mit maximaler x-Koordinate erstellt.

2. In der Liste wird aus allen Elementen, die noch keinen Wert S zugeordnet bekommen haben, das Start- (minimale z-Koordinate) und Stoppelement (maximale z-Koordinate) bestimmt.

3. Dem Startelement i_1 wird der Wert der zugehörigen Grobzelle $coarse[i_1]$ multipliziert mit v_1 zugeordnet.

$$S_{i_1} = v_1 T_{coarse[i_1]}$$

4. Man sucht die Nachbarn j_1, j_2 von i_1, für die gilt, $coarse[i] \neq coarse[j_1]$ bzw. $coarse[i] \neq coarse[j_2]$. Deren Wert der agglomerierten Zelle wird mit v_3 multipliziert und aufaddiert.

$$S_{i_1} = S_{i_1} + v_3 T_{coarse[j_1]} + v_3 T_{coarse[j_2]} \tag{4.1}$$

5. Dann bestimmt man den Nachbarn j_3 der Zelle j_1 und j_2, der Nachbar beider Zellen, aber nicht i ist. Der Werte der agglomerierten Zelle von

j_3 wird dann mit dem Faktor v_5 aufaddiert.

$$S_{i_1} = S_{i_1} + v_5 T_{coarse[j_3]}$$

6. Danach multipliziert man den Wert S noch mit dem Faktor $\frac{1}{f_P}$

$$S_{i_1} = \frac{1}{f_P} S_{i_1}$$

7. Mit den Nachbarn i_2 und i_3 von i_1, die die gleiche agglomerierte Zelle besitzen, wird das gleiche Prozedere aus den Schritten 3 bis 6 durchgeführt, aber mit i_2 bzw. i_3 statt i_1, v_2 statt v_1, v_4 statt v_3 und v_6 statt v_5 durchgeführt.

8. Für die vierte Unterzelle der agglomerierten Zelle $coarse[i_1]$, hier mit i_4 bezeichnet, wiederholt man die Schritte 3 bis 6 mit i_4 statt i_1

9. Die Zellen i_2 und i_3 werden auf ihre Nachbarn ohne einen Wert für S überprüft. Diese sind die möglichen Kandidaten für den nächsten Start. Dabei wird dasjenige Element gewählt, welches selbst genau zwei Nachbarn besitzt, die noch keinen Wert S erhalten haben. Wenn das Startelement i_1 noch das aus Schritt 2 ist, so trifft diese Bedingung auf zwei Elemente zu. Deshalb wird in diesem Fall das Element mit der kleineren x-Koordinate als neues Startelement gewählt.

10. Dann werden die Schritte 3 - 9 wiederholt, bis dem Stoppelement ein Wert S zugeordnet wurde.

11. Es wurden nun zwei Schichten bezüglich der z-Richtung bearbeitet. Um die verbleibenden Schichten zu bearbeiten werden die Schritte 2 bis 10 wiederholt, bis allen Elementen ein neuer Wert S zugeordnet wurde.

Als Beispiel sei folgendes Bild gegeben.

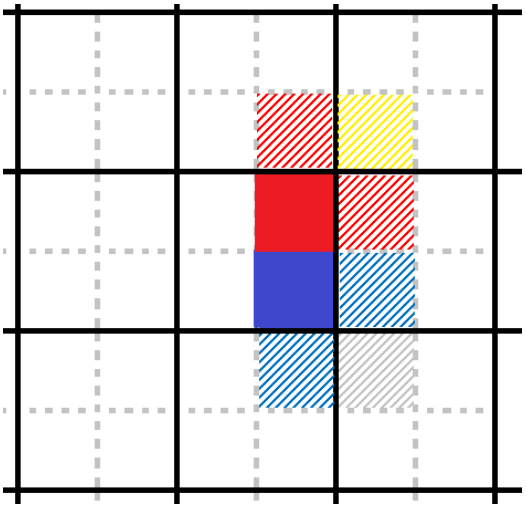


Abbildung 4.14: Beispiel der Prolongation

Der Wert des Startelements in blau ergibt sich als Summe der angrenzenden dick umrandeten Zellen. Die Gewichtung ist dabei für die umrandete Zelle mit der blauen Zelle v_1, für die umrandeten Zellen mit der blau schraffierten Zelle v_3 und für die agglomerierte Zelle der grau schraffierte Zelle v_5. Anschließend wird der Wert des Startelements mit $\frac{1}{f_P}$ multipliziert.
Für die rote Nachbarzelle ergibt sich der neue Wert wie folgt. Es ist gewichtete Summe aus der umrandeten Zelle, welche die rote Zelle enthält, mit Gewichtung v_2, aus den umrandeten Zellen mit der rot schraffierten Zelle mit Gewicht v_4 Zelle und der umrandeten Zelle, die die gelb schraffierte Zelle enthält, gewichtet mit v_6. Anschließend wird der Wert des Nachbarelements auch mit $\frac{1}{f_P}$ multipliziert.
Für die Restriktion geht man wie folgt vor. Seien S die gegebenen Werte des feinen Netzes und T die gesuchten Werte des groben Netzes. Außerdem sei durch $coarse[i]$ die zur Zelle i des feinen Netzes die dazugehörige agglomerierte Zelle des groben Netzes gegeben.

1. Es wird eine Liste der Elemente des feinen Netzes mit maximaler x-Koordinate erstellt.

2. In der Liste wird aus allen Elementen, die noch keinen Wert S zugeordnet bekommen haben, das Start- (minimale z-Koordinate) und

Stoppelement (maximale z-Koordinate) bestimmt.

3. Der agglomerierten Zelle des Startelements i_1 wird der Wert 0 zugeordnet, also $T_{coarse[i_1]} = 0$.

4. Der agglomerierten Zelle des Startelements i_1 wird der Wert von i_1 multipliziert mit v_1 zugeordnet.

$$T_{coarse[i_1]} = T_{coarse[i_1]} + v_1 S_{i_1}$$

5. Man sucht die Nachbarn j_1, j_2 von i_1, für die gilt, $coarse[i] \neq coarse[j_1]$ bzw. $coarse[i] \neq coarse[j_2]$. Deren Wert wird mit v_3 multipliziert und aufaddiert.

$$T_{coarse[i_1]} = T_{coarse[i_1]} + v_3 S_{i_2} + v_3 S_{i_3} \tag{4.2}$$

6. Dann bestimmt man den Nachbarn j_3 der Zelle j_1 und j_2, der Nachbar beider Zellen, aber nicht i ist. Der Werte von j_3 wird dann mit dem Faktor v_5 aufaddiert.

$$T_{coarse[i_1]} = T_{coarse[i_1]} + v_5 S_{i_3}$$

7. Mit den Nachbarn i_2 und i_3 von i_1, die die gleiche agglomerierte Zelle besitzen, wird das gleiche Prozedere aus den Schritten 4 bis 6 durchgeführt, aber mit i_2 bzw. i_3 statt i_1, v_2 statt v_1, v_4 statt v_3 und v_6 statt v_5 durchgeführt.

8. Für die vierte Unterzelle der agglomerierten Zelle $coarse[i_1]$, hier mit i_4 bezeichnet, wiederholt man die Schritte 4 bis 6 mit i_4 statt i_1

9. Danach multipliziert man den Wert T noch mit dem Faktor $\frac{1}{4f_P}$

$$T_{coarse[i_1]} = \frac{1}{4f_P} T_{coarse[i_1]}$$

10. Die Zellen i_2 und i_3 werden auf ihre Nachbarn überprüft. Diejenigen, deren zugehörige agglomerierte Zelle noch keinen Wert T zugeordnet

bekommen haben, sind mögliche Kandidaten für den nächsten Start. Dabei wird dasjenige Element gewählt, welches selbst genau zwei Nachbarn besitzt, deren agglomerierte Zelle noch keinen Wert T erhalten haben. Wenn das Startelement i_1 noch das aus Schritt 2 ist, so trifft diese Bedingung auf zwei Elemente zu. Deshalb wird in diesem Fall das Element mit der kleineren x-Koordinate als neues Startelement gewählt.

11. Dann werden die Schritte 3 - 10 wiederholt, bis der Grobzelle des Elements Stopp ein Wert T zugeordnet wurde.

12. Es wurden nun zwei Schichten bezüglich der z-Richtung bearbeitet. Um die verbleibenden Schichten zu bearbeiten werden die Schritte 2 bis 11 wiederholt, bis alle Elemente einen neuen Wert T erhalten haben.

Pseudocode für die Restriktion bzw. die Prolongation findet sich im Anhang.

Kapitel 5

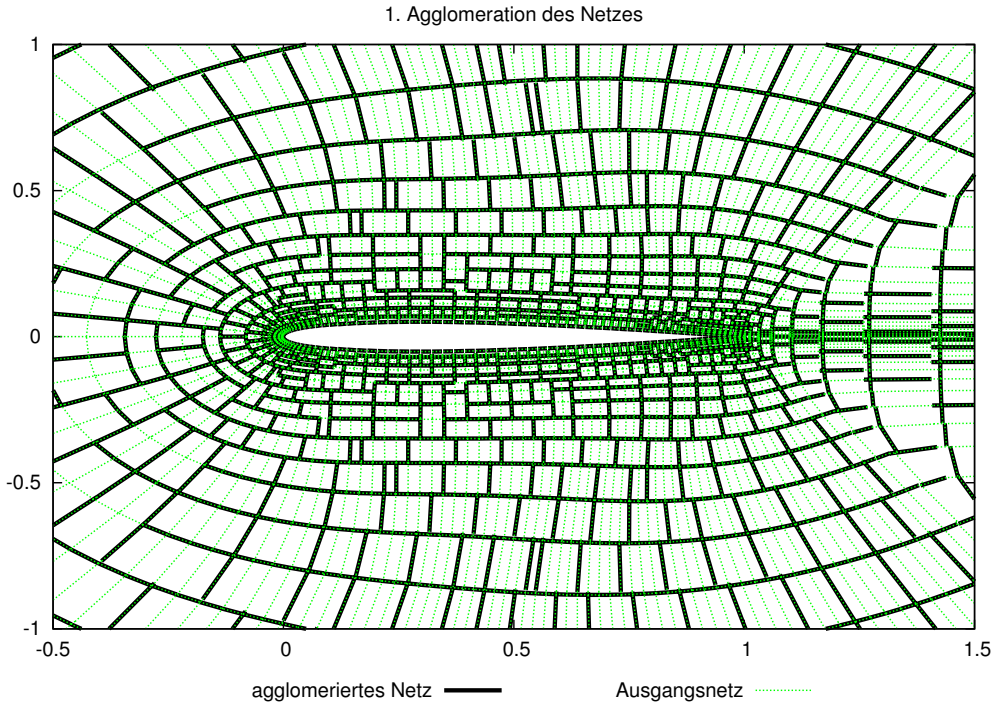
Ergebnisse

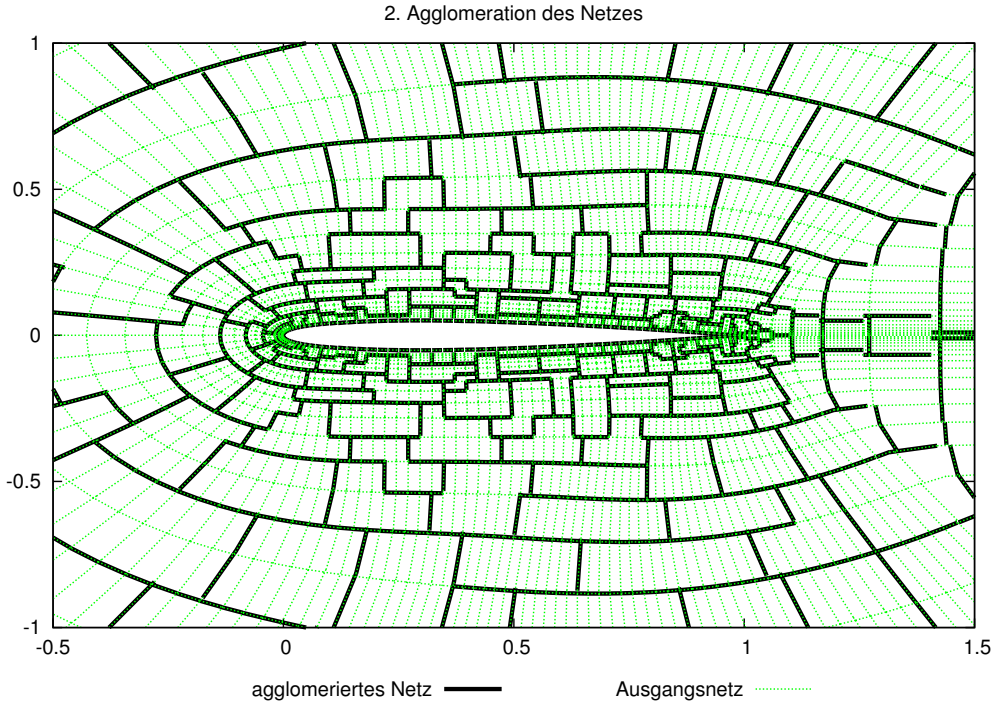
Die folgenden Ergebnisse beruhen alle auf einem Netz um den Trägerquerschnitts NACA 0012. Die Netzgröße bewegt sich dabei zwischen 160×32 Elemente bis 512×128 Elemente. Die Parameter wurden wie in folgender Tabelle dargestellt gewählt.

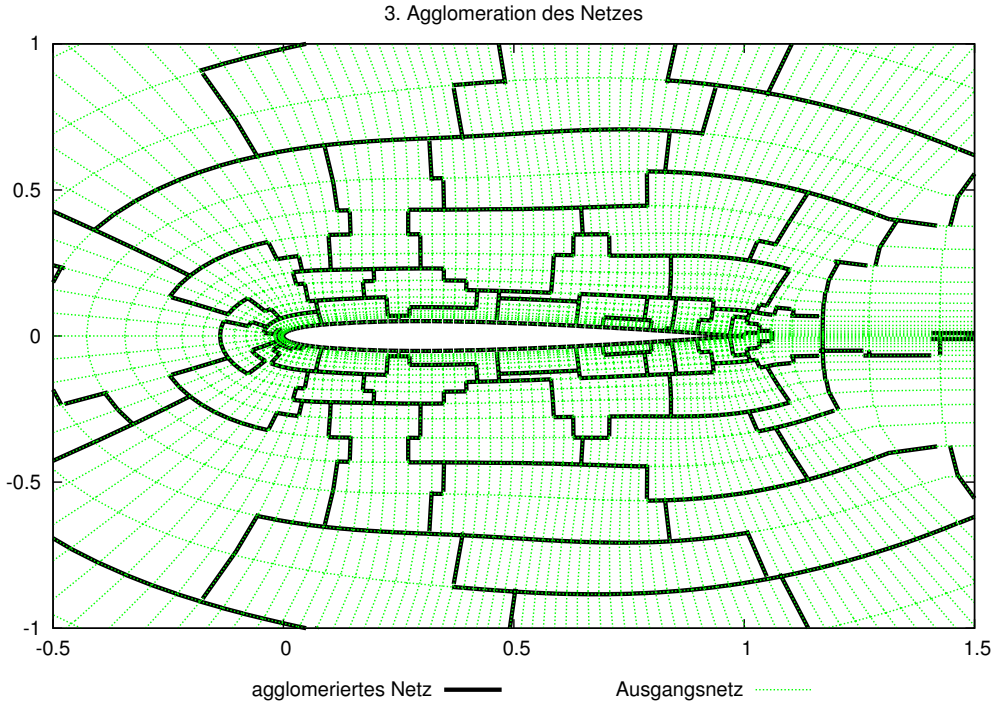
Parameter	**Euler-Gleichung**	**laminare Navier-Stokes-Gleichung**
Referenz-Mach-Zahl	0,8	0,5
Referenz-Temperatur	273,15	273,15
Referenz-Druck	101325	101325
Gaskonstante	287	287
Wärmekapazitätskoeffizient	1,4	1,4
Sutherland-Konstante	110,4	110,4
Laminare Prandtl-Zahl	0,72	0,72
Turbulente Prandtl-Zahl	0,9	0,9
Anstellwinkel	je nach Testfall	je nach Testfall
Reynolds-Zahl	$6,5 * 10^6$	5000
Start-CFL-Zahl	3	10
Finale CFL-Zahl	1000	1000
γ	5	5
Mehrgitterzyklus	4w	4w

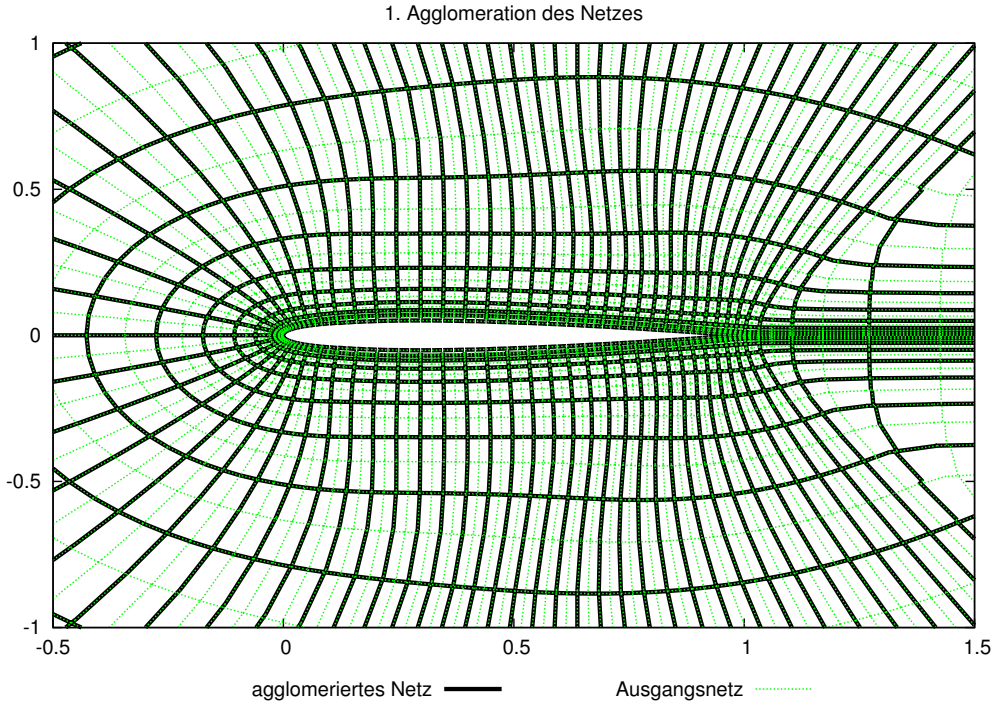
Tabelle 5.1: Parameter nach Art

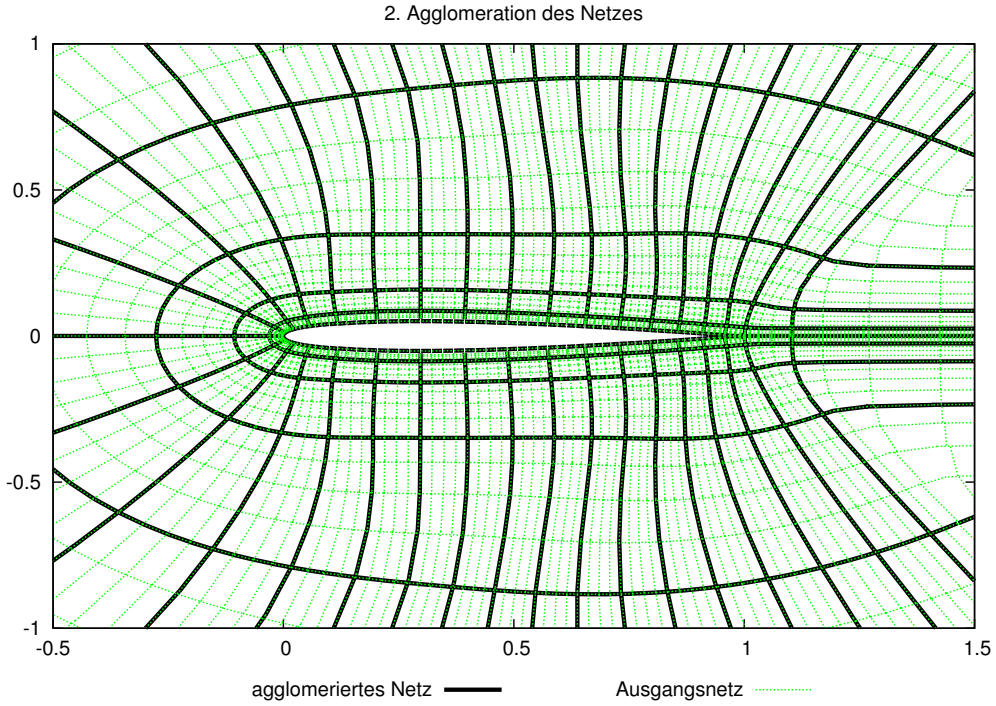
Zunächst soll der Unterschied in der Agglomeration gezeigt werden. Die 3 ersten Bilder zeigen dabei die Agglomeration mittels MGridgen, während die anderen drei den neuen Algorithmus zeigen.

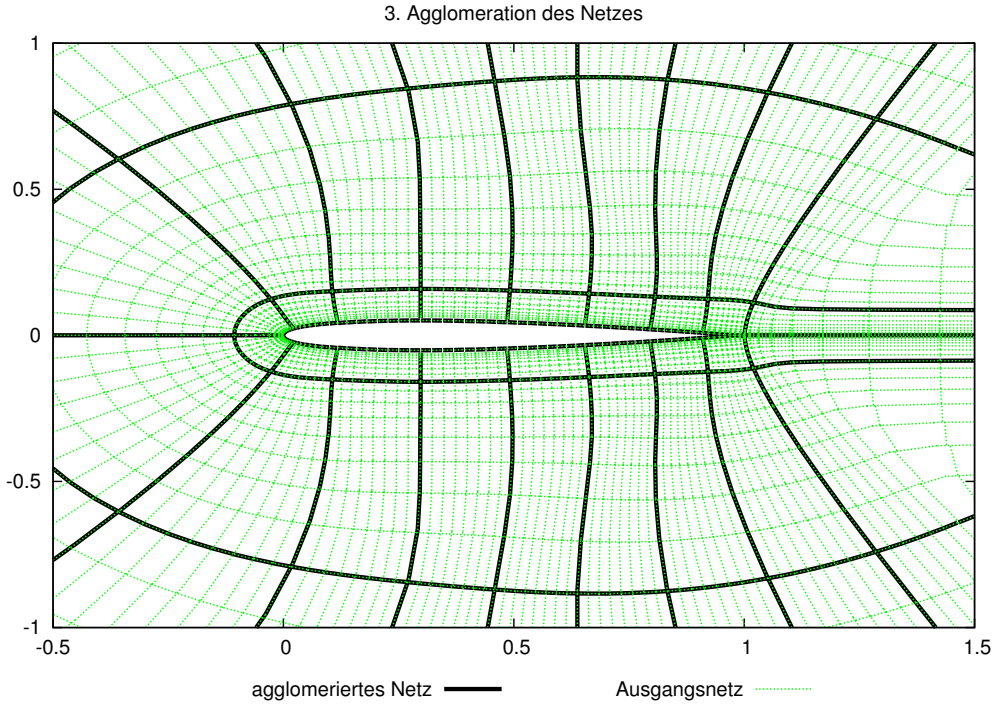



2. Agglomeration des Netzes
1
0.5
0
-0.5
-1
-0.5
0
0.5
1
1.5
agglomeriertes Netz
Ausgangsnetz


3. Agglomeration des Netzes
1
0.5
0
-0.5
-1
-0.5
0
0.5
1
1.5
agglomeriertes Netz
Ausgangsnetz


1. Agglomeration des Netzes
1
0.5
0
-0.5
-1
-0.5
0
0.5
1
1.5
agglomeriertes Netz
Ausgangsnetz


2. Agglomeration des Netzes
1
0.5
0
-0.5
-1
-0.5
0
0.5
1
1.5
agglomeriertes Netz
Ausgangsnetz



Das neue Verfahren generiert die gewünschte Struktur. Man erkennt vor allem in der gröbsten Agglomeration, dass mittels des volumenbasierten Verfahrens die Symmetrie und die ähnliche Form der Elemente deutlich verloren geht. MGridGen agglomeriert aber dennoch einheitlich geformte Netze.

5.1 Euler-Gleichungen auf dem 160×32 Netz

Die Ergebnisse der Euler-Gleichungen auf den 160×32 Netz sind wie folgt.

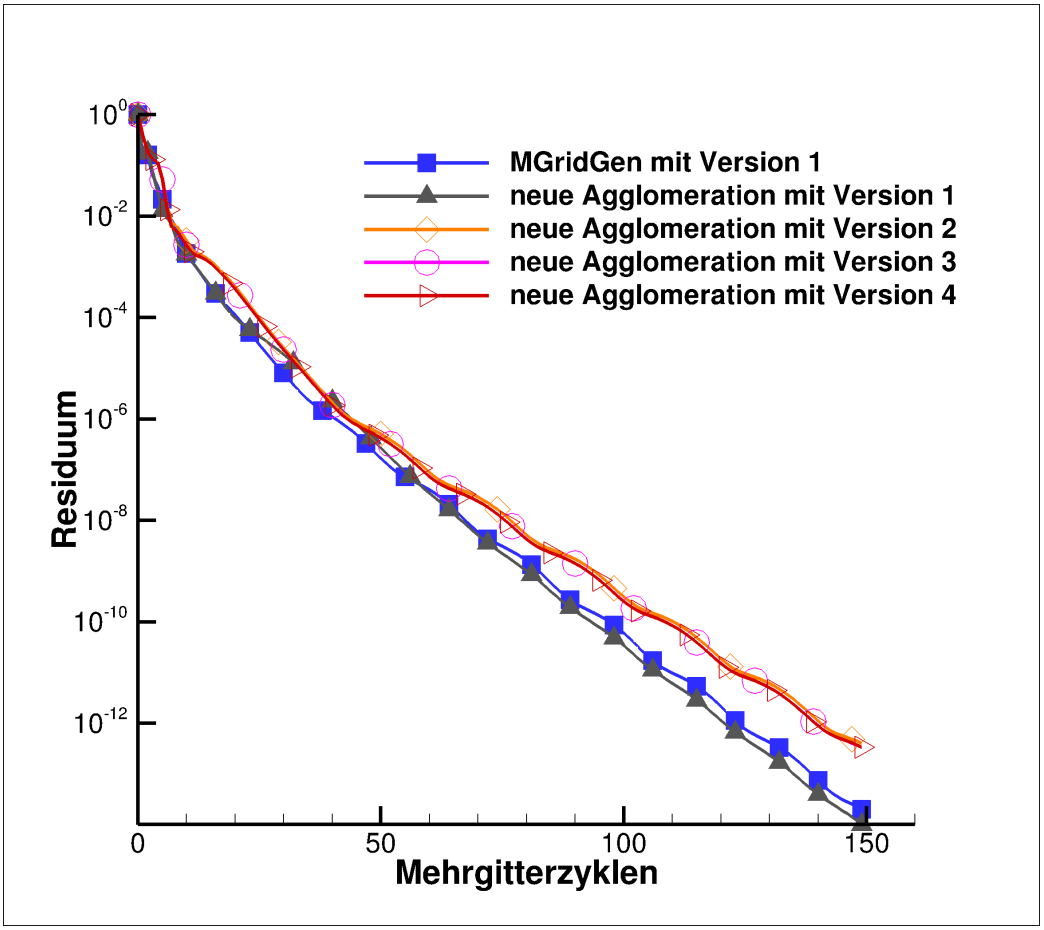


Abbildung 5.1: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von $1,25°$

Hierbei steht Variante 1 für die Injektion als Prolongation und die volumengewichtete Interpolation als Restriktion, Variante 2 für die Prolongation und Restriktion von Wesseling, 3 für ein bilineares Verfahren und 4 für die Methode von Kwak. Man erkennt, dass die neue Agglomeration minimal besser ist, aber die Veränderung der Operatoren führte zu keiner Verbesserung. Es führen aber alle Varianten zur einer Konvergenz auf dem kleinsten Testfall.

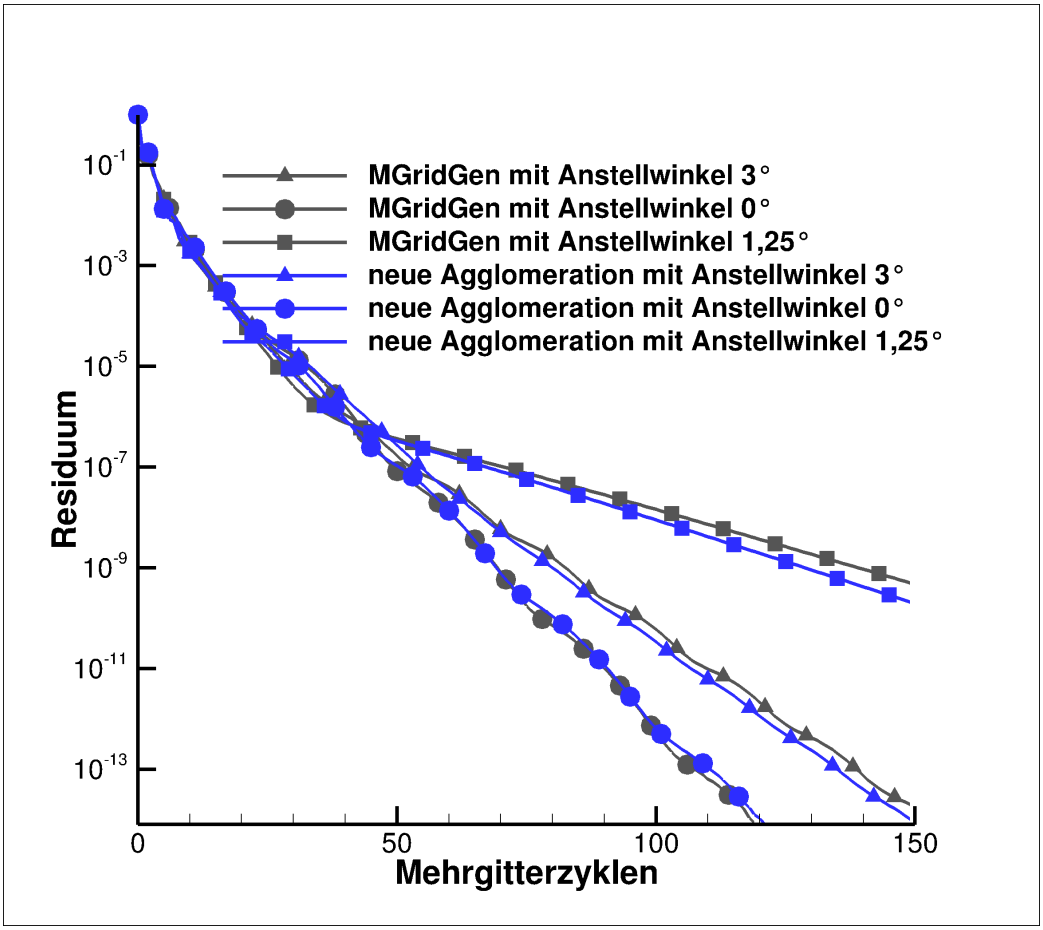


Abbildung 5.2: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren unter Verwendung der Version 1 und unterschiedlichen Anstellwinkeln

Die beiden Agglomerationsverfahren unterscheiden sich nicht stark bei gleichem Anstellwinkel. Im symmetrischen Testfall von 0° ist das Verfahren mit MGridGen sogar leicht besser.

5.2 Euler-Gleichungen auf dem 320×64 Netz

Der Testfall auf dem 320×64-elementigen Netz erzeugt folgende Ergebnisse.

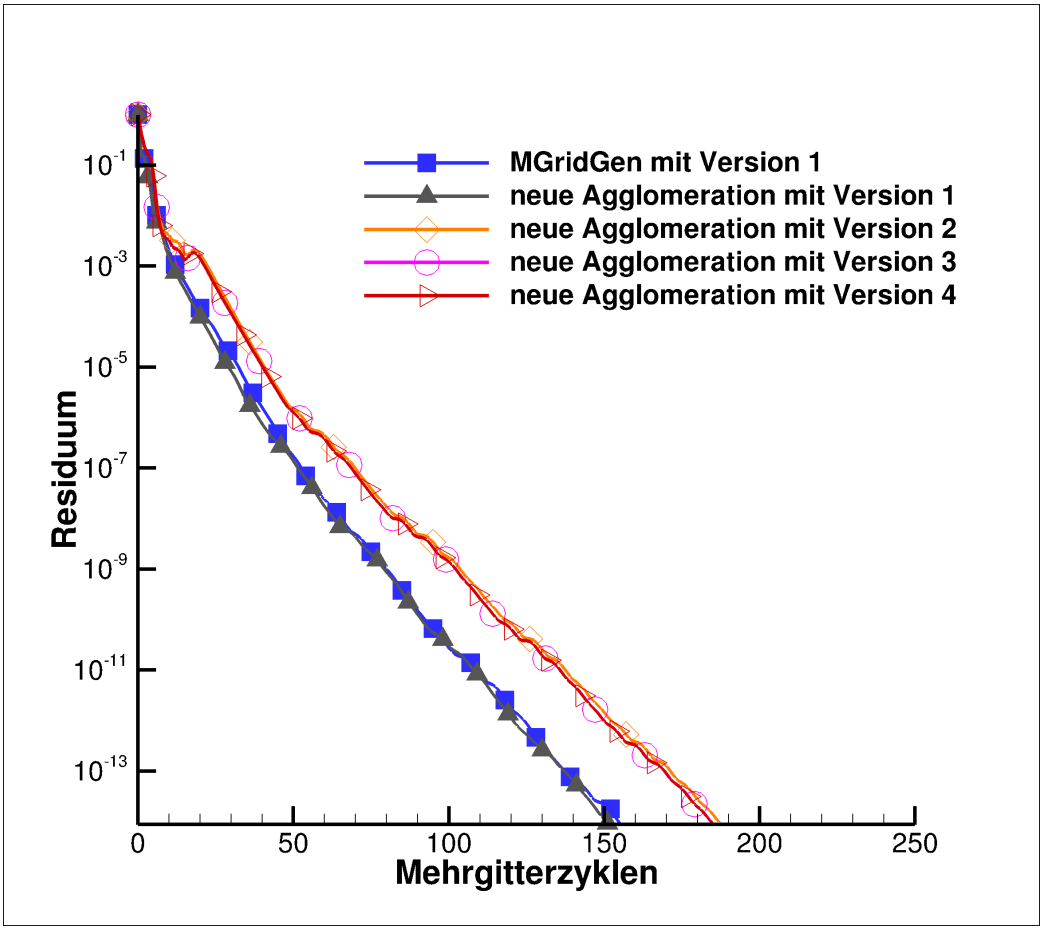


Abbildung 5.3: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von $1,25°$

Auch auf dem größeren Netz ergibt sich ein ähnliches Bild wie im Fall auf dem 160×32-Netz. Die Version mit der Injektion und volumengewichteten Interpolation konvergiert am schnellsten. Die anderen Versionen führen zu einer leichten Verschlechterung.

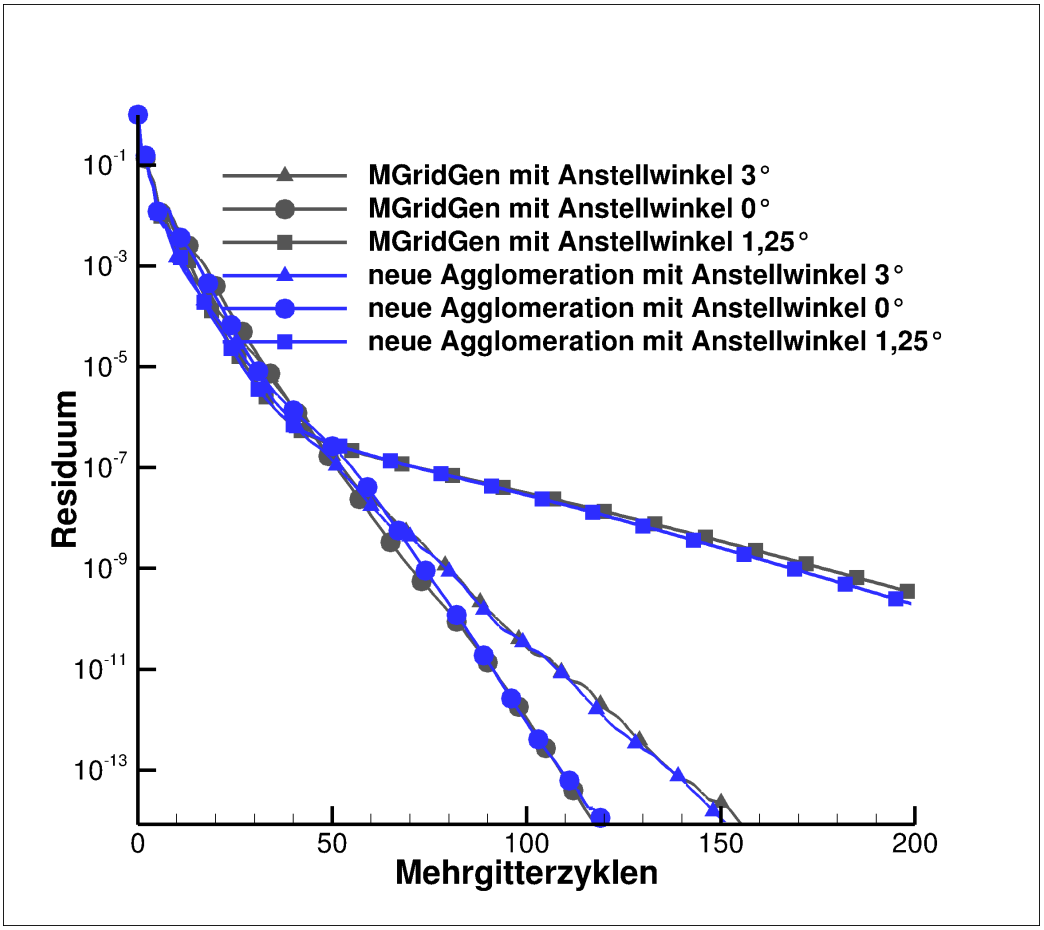


Abbildung 5.4: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren unter Verwendung der Version 1 und unterschiedlichen Anstellwinkeln

Im Vergleich der beiden Agglomerationsverfahren erkennt man wieder, dass der Anstellwinkel auf diesem Testfall eine deutlich größere Rolle spielt, als das Verfahren selbst.

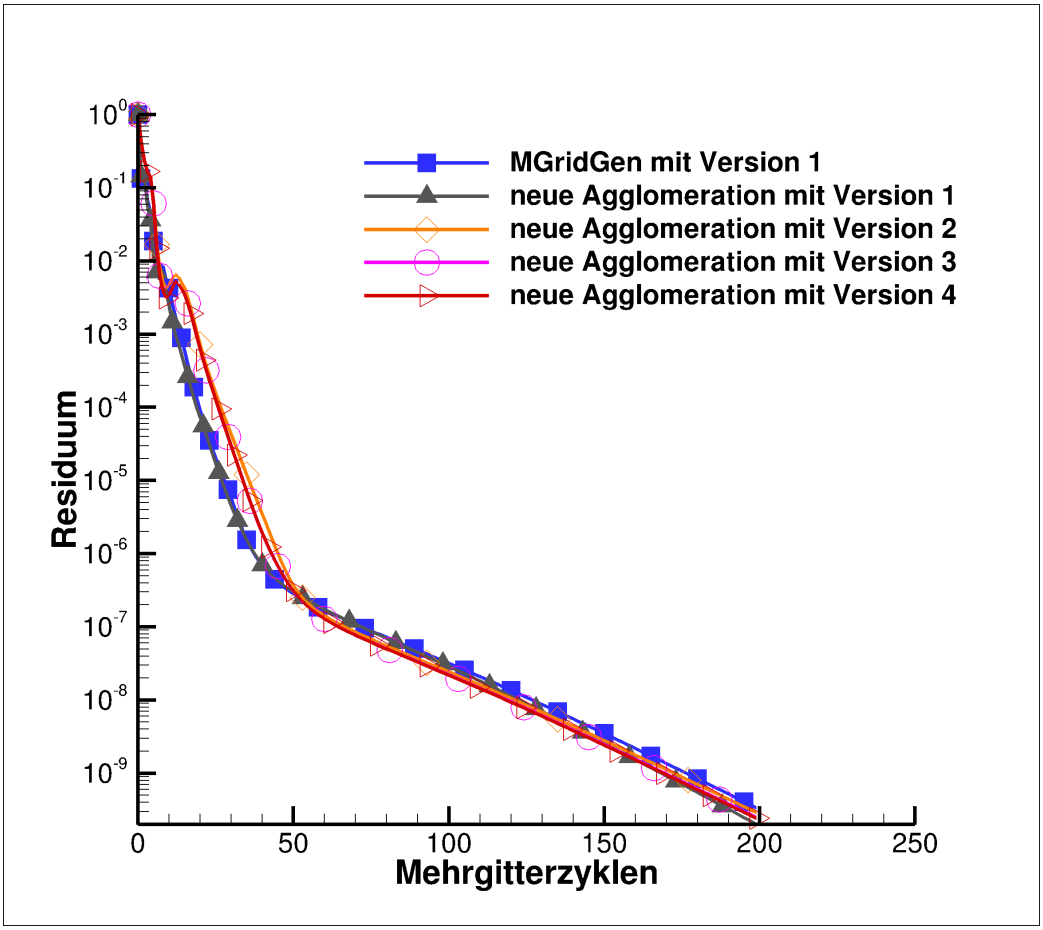


Abbildung 5.5: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von 0°

Im Fall eines Anstellwinkels von 0° erkennt man, dass sich alle Varianten ungefähr gleich gut schlagen. Im Gegensatz dazu waren im vorherigen Fall mit $1,25°$ die neuen Versionen erkennbar schlechter.

5.3 laminare Navier-Stokes-Gleichungen auf dem 128×32 Netz

Nun sollen die laminaren Navier-Stokes-Gleichungen auf einem Netz der Größe 128×32 dargestellt werden:

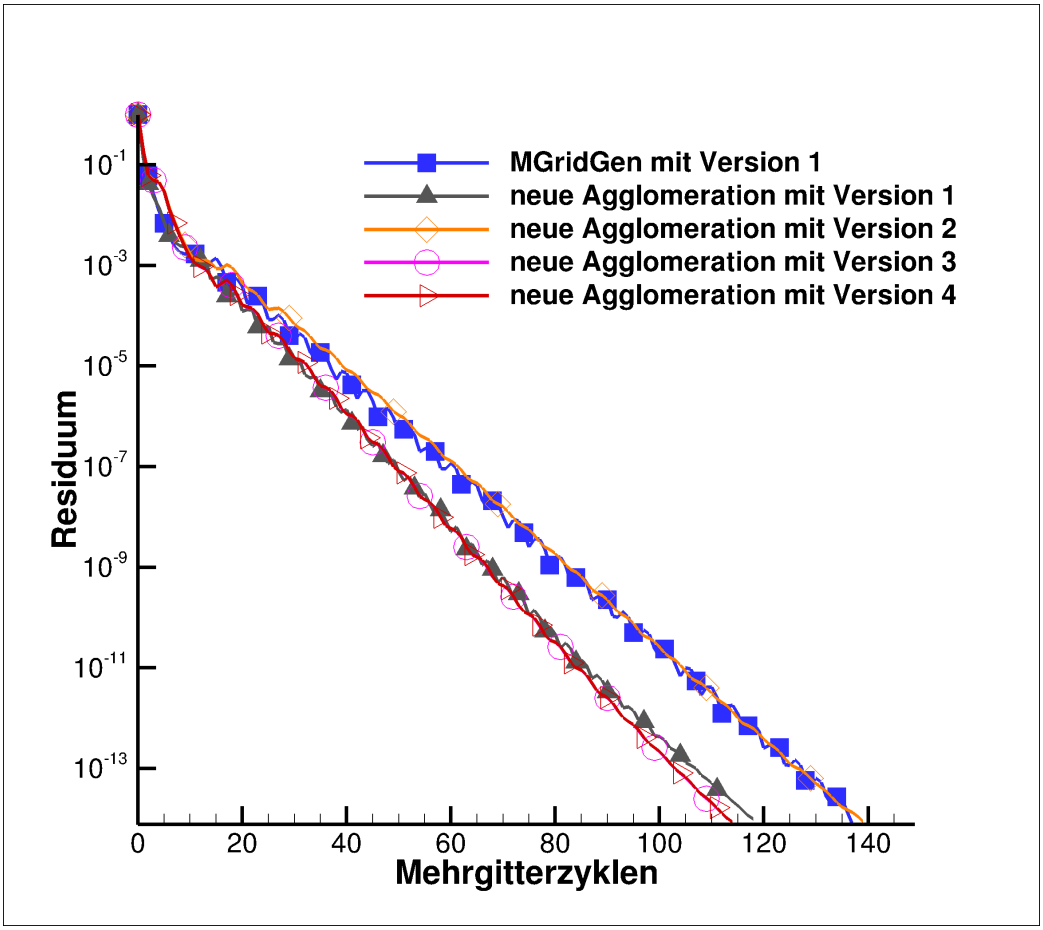


Abbildung 5.6: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von 2°

Bei einem Anstellwinkel von 2° ergibt sich, dass MGridGen und die neue Agglomeration mit dem Verfahren von Wesseling zu den schlechtesten Ergebnissen führt. Die anderen 3 Varianten des neuen Algorithmus konvergieren etwas schneller.
Für die Variante mit einem Anstellwinkel von 0° ergibt sich hingegen ein anderes Bild. Hierbei ist zwar auch das Verfahren von Wesseling am schlechtesten, aber das MGridGen-Verfahren ist wieder genauso gut wie die anderen Varianten der neuen Agglomeration.

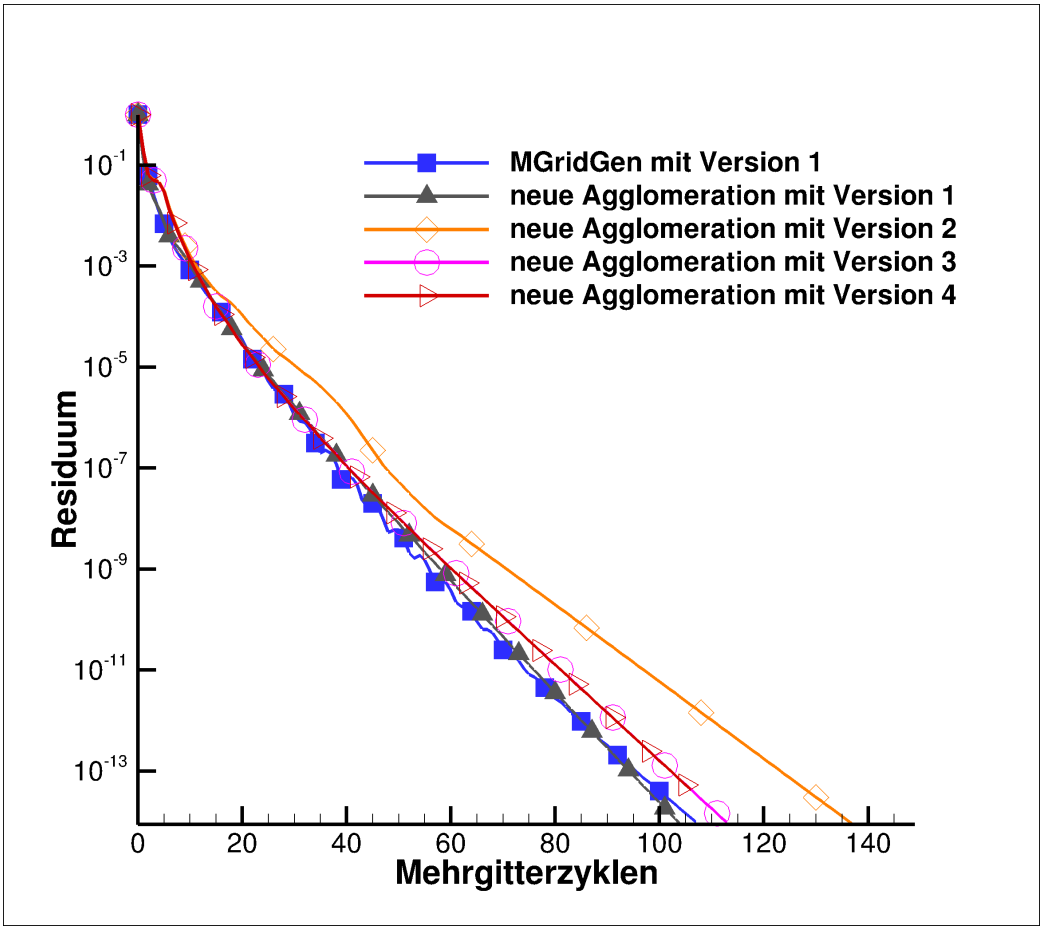


Abbildung 5.7: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von 0°

Vergleicht man MGridGen und die neue Agglomeration mit der gleichen Prolongation und Restriktion, so erkennt man, dass in diesem Fall das MGridGen-Verfahren sichtbar schlechter ist. Lediglich bei einem Anstellwinkel von 2° unterscheiden sich die beiden Verfahren kaum.

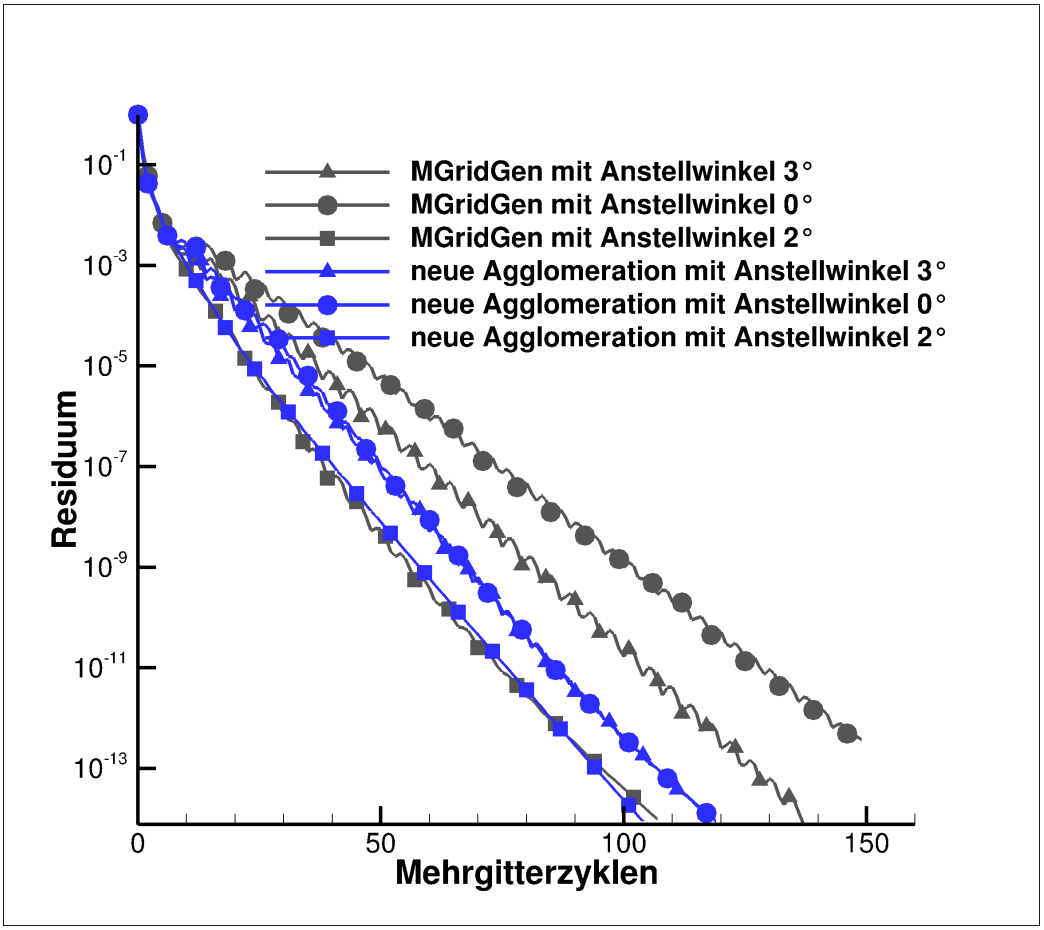


Abbildung 5.8: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren unter Verwendung der Version 1 und unterschiedlichen Anstellwinkeln

5.4 laminare Navier-Stokes-Gleichungen auf dem 512×128 Netz

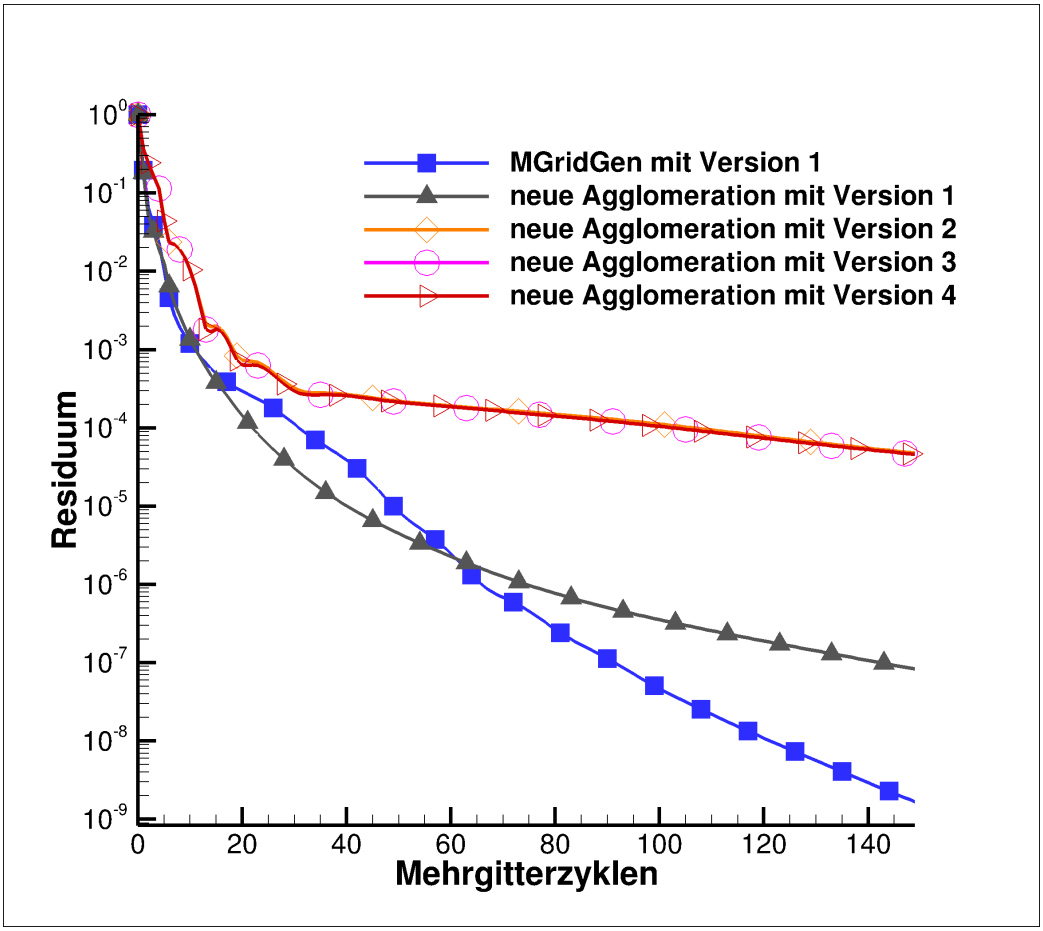


Abbildung 5.9: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von 1°

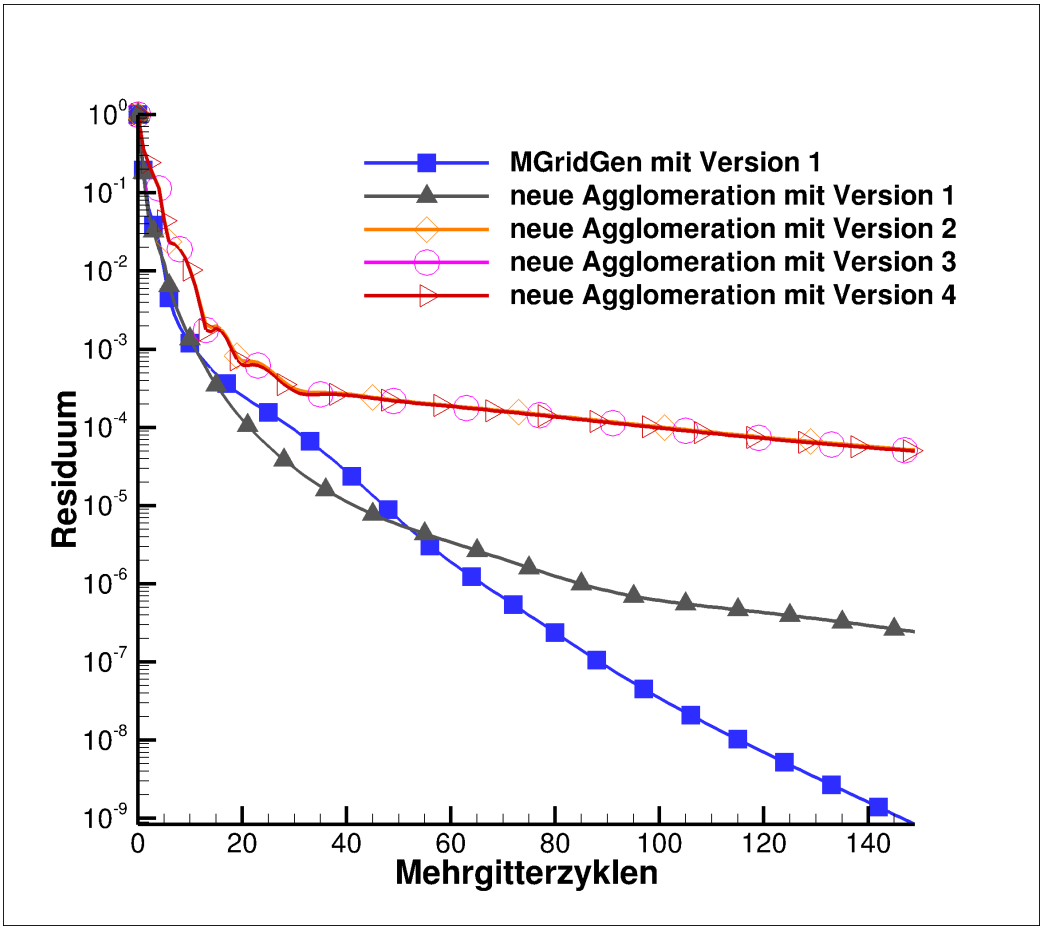


Abbildung 5.10: Vergleich des MgridGen-Verfahrens mit dem neuen Verfahren und verschiedenen Operatoren bei einem Anstellwinkel von 0°

Auf dem 512×128 Netz ergibt sich eine etwas andere Auswertung. Sowohl im Fall von einem Anstellwinkel von 0° als auch bei 1° ist das MGridGen-Verfahren deutlich besser als alle andere Verfahren. Die neuen Operatoren sind zudem deutlich schlechter als die neue Agglomeration mit der Version 1.

Im Anhang befinden sich zudem Bilder, welche auf den agglomerierten Netzen erzeugt wurden. Da die Auswertung der Druckverteilung mittels c_P-Verteilung oder der Strömungsverlauf aber nicht Teil der Auswertung sein sollen, soll hier nur als Beispiele für mögliche Bilder, die mit den Ergebnissen erzeugt werden können, darauf verwiesen werden.

Kapitel 6

Fazit und Ausblick

In der vorliegenden Arbeit wurde ein neues Verfahren entwickelt um auf unstrukturierten Netzen mit bestimmten Eigenschaften eine Grobgittersequenz ähnlich der im strukturierten Fall zu erzeugen. Desweiteren wurden Operatoren aus den strukturierten Netzen auf die neue Sequenz angewendet. Dies erfolgte durch Programmierung eines neuen Teilprogramms in den vorhandenen TAU-Code. Diese Verfahren wurden dann auf vier ausgewählten Testfällen mit verschiedenen Anstellwinkeln getestet.
Wie man in den Ergebnissen erkennt, gibt es wenig Unterschiede im Konvergenzverhalten der bereits implementierten Agglomerationsstrategie mit MGridGen und der neuen Methode für die Testfälle mit den Euler-Gleichungen. Auch die Hinzunahme von Operatoren aus dem Bereich der strukturierten Netze führt zu keiner Verbesserung der Konvergenz. Eine mögliche Erklärung ist, dass das Mehrgitterverfahren ursprünglich aus sehr einfachen und mit einer geringen Anzahl an Rechenschritten ausführbaren Operationen bestand. Durch den Fortschritt auf dem Gebiet der Mehrgitterverfahren wurden aber immer komplexere und deutlich aufwändigere Verfahren kombiniert. Diese können daher wahrscheinlich eher schlechter durch die relativ einfachen Modifikationen in Agglomeration und Operatoren stark beeinflusst werden.
Für die laminaren Navier-Stokes-Gleichungen erkennt man jedoch, dass auf dem Netz mit 128×32 eine Verbesserung für bestimmte Anstellwinkel erkennbar ist, während für das 512×128-Netz eine leichte Verschlechterung für

die neue Methode und eine deutlich schlechtere Konvergenz unter Hinzunahme anderer Operatoren erkennbar ist. Eine mögliche Erklärung ist, dass die deutlich komplizierteren Operatoren zu einer stärkeren Glättung des Residuums als die Standardverfahren führen. Da der Fehler aber durch den an das Runge-Kutta-Verfahren angelehnten Glätter aber an sich schon stark geglättet wird, führen diese neuen Operatoren vielleicht eher zu einer Überglättung und damit Verschlechterung des Konvergenzverhaltens.
Insgesamt kann man aber feststellen, dass das Verhalten sehr vom Testfall abhängt. Zudem führt die neue Agglomerationsmethode zwar meist nicht zu einer Verbesserung, aber es ist auch in den meisten Fällen keine deutliche Verschlechterung erkennbar.
Es wäre interessant in einer weiteren Arbeit zu untersuchen, woher diese Unterschiede im Verhalten stammen und ob man daraus Merkmale ableiten kann, für die sich das neue Verfahren besonders gut oder schlecht eignet.
Auch die Anwendung der neuen Methode auf weitere Testfälle mit anderen Parametern oder die Verwendung von turbulenten Testfällen ist von Interesse. Ein weiterer Testfall, der aber leider nicht in Rahmen dieser Arbeit untersucht werden konnte, sind Netze mit gemischten Strukturen, wie in 3.5 dargestellt. Eine Möglichkeit wäre es darauf in Form der hier vorgestellten neuen Methode zu arbeiten.
Zudem wäre die Anwendung auf Testfälle, von denen man weiß, dass die strukturierten Netze eine bessere Konvergenz aufweisen, auch interessant. Ein solcher Fall ist beispielsweise in der Einleitung beschrieben.
Abschließend kann man sagen, dass sich alle Ergebnisse nur auf eine kleine Zahl an Testfällen beschränkt und deshalb noch weitere Forschung nötig ist um weitere Fälle zu untersuchen. Die vorliegenden Ergebnisse sind dabei ein Hinweis darauf, dass die neue Methode in ihrer jetzigen Form zu keiner deutlichen Verbesserung führt, es aber eine ähnliche gute Alternative zu bisherigen Agglomerationsverfahren sein könnte.

Literaturverzeichnis

[1] Steven R Allmaras and Forrester T Johnson. Modifications and clarifications for the implementation of the spalart-allmaras turbulence model. In *Seventh international conference on computational fluid dynamics (ICCFD7)*, pages 1–11, 2012.

[2] Nikolai Sergeevich Bakhvalov. On the convergence of a relaxation method with natural constraints on the elliptic operator. *USSR Computational Mathematics and Mathematical Physics*, 6(5):101–135, 1966.

[3] Jiri Blazek. *Computational fluid dynamics: principles and applications*. Butterworth-Heinemann, 2015.

[4] BMU. Luftverkehr. `https://www.bmu.de/themen/luft-laerm-verkehr/verkehr/flugverkehr/`.

[5] Achi Brandt. Multi-level adaptive solutions to boundary-value problems. *Mathematics of computation*, 31(138):333–390, 1977.

[6] Haim Brezis. *Functional analysis, Sobolev spaces and partial differential equations*. Springer Science & Business Media, 2010.

[7] Dieter Etling. *Theoretische Meteorologie: Eine Einführung*. Springer-Verlag, 2008.

[8] Lawrence C. Evans. *Partial Differential Equations second edition*. American Mathematical Society, 2010.

[9] Radii Petrovich Fedorenko. A relaxation method for solving elliptic difference equations. *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, 1(5):922–927, 1961.

[10] Charles L Fefferman. Existence and smoothness of the navier-stokes equation. *The millennium prize problems*, 57:67, 2006.

[11] O Forster. Analysis 2, friedr. *Vieweg & Sohn, Braunschweig, Wiesbaden, Germany*, 1996.

[12] Otto Forster. Analysis 3, 3. *Auflage. Vieweg Studium*, 52, 1984.

[13] Otto Forster. *Analysis 1*, volume 12. Springer, 2004.

[14] Klaus Gersten and Heinz Herwig. *Strömungsmechanik: Grundlagen der Impuls-, Wärme-und Stoffübertragung aus asymptotischer Sicht*. Springer-Verlag, 2013.

[15] Wolfgang Hackbusch. *Multi-grid methods and applications*, volume 4. Springer Science & Business Media, 2013.

[16] Prof. Dr. R. Herzog. Skript zur Vorlesung Einführung in Sobolevräume. `https://www.tu-chemnitz.de/mathematik/part_dgl/teaching/WS2011_Optimale_Steuerung_PDEs/Skript_Sobolevr%C3%A4ume.pdf`.

[17] Antony Jameson, Wolfgang Schmidt, and Eli Turkel. Numerical solution of the euler equations by finite volume methods using runge kutta time stepping schemes. In *14th fluid and plasma dynamics conference*, page 1259, 1981.

[18] Volker John. Schwache Lösungstheorie. `https://www.wias-berlin.de/people/john/LEHRE/NUM_KONV_PROB/num_konv_prob_4.pdf`.

[19] Volker John. Multigrid methods. Technical report, Technical report, 2013.

[20] Juha Kinnunen. Sobolev spaces. *Department of Mathematics and Systems Analysis, Aalto University*, 2017.

[21] Norbert Köckler. *Mehrgittermethoden: Ein Lehr-und Übungsbuch.* Springer-Verlag, 2012.

[22] N. Kroll, M. Abu-Zurayk, D. Dimitrov, T. Franz, T. Führer, T. Gerhold, S. Görtz, R. Heinrich, C. Ilic, J. Jepsen, J. Jägersküpper, M. Kruse, A. Krumbein, S. Langer, D. Liu, R. Liepelt, L. Reimer, M. Ritter, A. Schwöppe, J. Scherer, F. Spiering, R. Thormann, V. Togiti, D. Vollmer, and J.-H. Wendisch. DLR project Digital-X: towards virtual aircraft design and flight testing based on high-fidelity methods. *CEAS Aeronautical Journal*, 7(1):3–27, 2016.

[23] Do Y Kwak. V-cycle multigrid for cell-centered finite differences. *SIAM Journal on Scientific Computing*, 21(2):552–564, 1999.

[24] Stefan Langer. Agglomeration multigrid methods with implicit runge–kutta smoothers applied to aerodynamic simulations on unstructured grids. *Journal of Computational Physics*, 277:72–100, 2014.

[25] Stefan Langer. *Preconditioned Newton methods to approximate solutions of the Reynolds averaged Navier-Stokes equations.* DLR, 2018.

[26] Stefan Langer, Axel Schwöppe, and Norbert Kroll. The DLR Flow Solver TAU - Status and Recent Algorithmic Developments. In *Proc. 52nd Aerospace Sciences Meeting, January 2014*, number 2014-0080 in Conference Proceeding Series. AIAA, 2014.

[27] Stefan Langer, Axel Schwöppe, and Norbert Kroll. Investigation and Comparison of Implicit Smoothers Applied in Agglomeration Multigrid. *AIAA Journal*, 53(8):2080 – 2096, 2015.

[28] Jean Leray et al. Sur le mouvement d'un liquide visqueux emplissant l'espace. *Acta mathematica*, 63:193–248, 1934.

[29] Prof. Dr. Thomas Lorenz. Skript zur Vorlesung Partielle Differentialgleichungen, Wintersemester 2018/19.

[30] Marcus Mohr and Roman Wienands. Cell-centred multigrid revisited. *Computing and Visualization in Science*, 7(3-4):129–140, 2004.

[31] Irene Moulitsas and George Karypis. Mgridgen parmgridgen. *Serial/Parallel Library for Generating Coarse Grids for Multigrid Methods,*, 2001.

[32] Prof. Dr.-Ing. Stefan Rill MSc. Aerodynamik des flugzeugs - massenerhaltung - kontinuität. `http://homepages.hs-bremen.de/~kortenfr/Aerodynamik/script/node7.html`.

[33] Rolf Rannacher. Numerische mathematik 2 (numerik partieller differentialgleichungen). *Lecture Notes, Heidelberg University, http://numerik. uni-hd. de/ lehre/notes*, 2008.

[34] Rolf Rannacher. Numerische mathematik 1 (numerik gewöhnlicher differentialgleichungen). *Lecture Notes, Heidelberg University, http://numerik. uni-hd. de/ lehre/notes*, 2017.

[35] Philip L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of computational physics*, 43(2):357–372, 1981.

[36] Philip L Roe. Characteristic-based schemes for the euler equations. *Annual review of fluid mechanics*, 18(1):337–365, 1986.

[37] Martin H Sadd. *Elasticity: theory, applications, and numerics*. Academic Press, 2009.

[38] Hermann Schlichting and Klaus Gersten. *Grenzschicht-theorie*. Springer-Verlag, 2006.

[39] R Charles Swanson, Eli Turkel, and C-C Rossow. Convergence acceleration of runge–kutta schemes for solving the navier–stokes equations. *Journal of Computational Physics*, 224(1):365–388, 2007.

[40] RC Swanson, Eli Turkel, and C-C Rossow. Convergence acceleration of runge–kutta schemes for solving the navier–stokes equations. *Journal of Computational Physics*, 224(1):365–388, 2007.

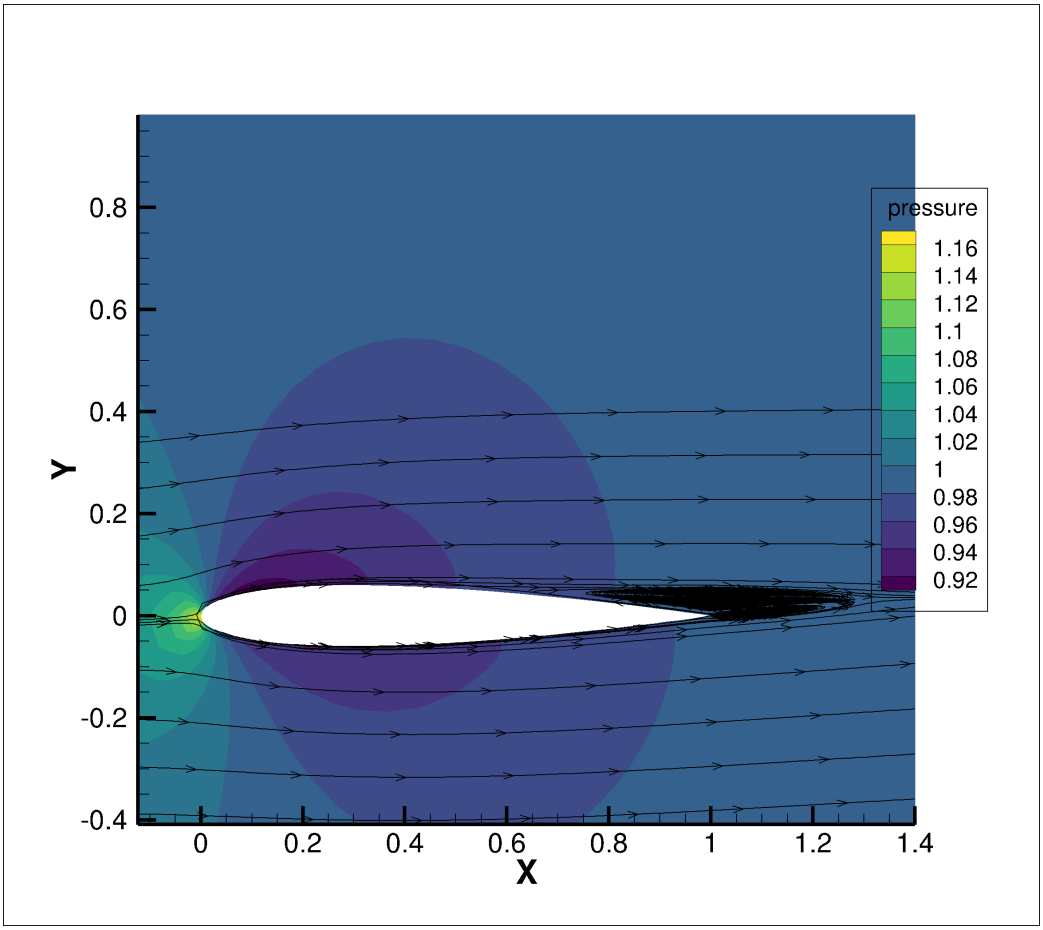
[41] Ulrich Trottenberg, Cornelius W Oosterlee, and Anton Schuller. *Multigrid*. Elsevier, 2000.

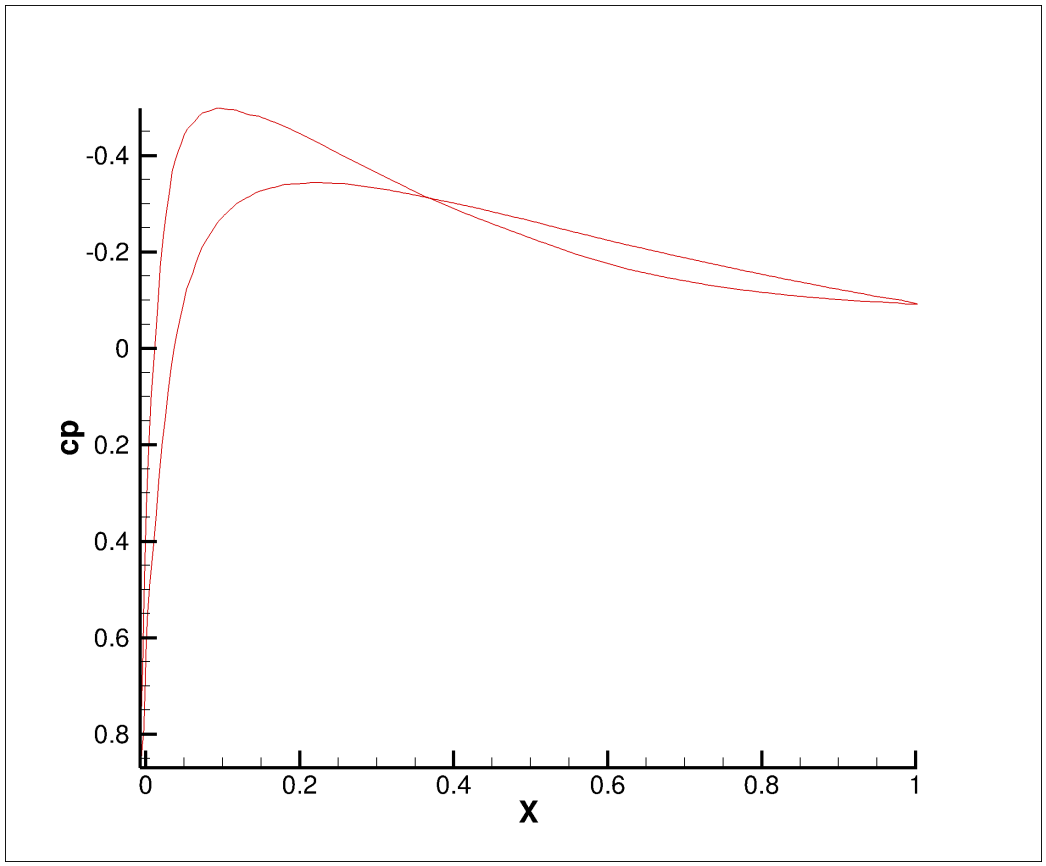
[42] Pieter Wesseling. Introduction to multigrid methods. 1995.

[43] Jörg Wolf. Partielle Differentialgleichungen I. `https://www.math.hu-berlin.de/~jwolf/web/GrunglagenPDE-Teil2.pdf`.

Anhang A

Bilder




-0.4
-0.2
0
0.2
0.4
0.6
0.8
cp
0
0.2
0.4
0.6
0.8
1
X

Anhang B

Pseudocode

Algorithm 3 Hilfsfunktion zur Prolongation

```
1: procedure PROLONG_HELP(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse, zelle, v_i, v_j, v_l)
2:     S[zelle]=v_i*T[coarse[zelle]]
3:     for jede Nachbarzelle j der Zelle zelle do
4:         if coarse[j]≠ coarse[zelle] then
5:             S[zelle]=S[zelle]+v_j*T[coarse[j]]
6:             j in list_neigh_help aufnehmen
7:         end if
8:     end for
9:     for jede Zelle j in list_neigh_help do
10:        jede Nachbarzelle k der Zelle j ungültig in list_neigh_2_help aufnehmen
11:    end for
12:    doppelt auftretende Zelle k mit k≠zelle in list_neigh_2_help finden und S[zelle]=S[zelle]+v_l*T[coarse[k]] setzen
13:    S[zelle]=(1/f_P)S[zelle]
14:    return S[zelle]
15: end procedure
```

Algorithm 4 Prolongation

```
1: procedure PROLONGATION(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse)  ▷ Wendet den Prolongationsoperator auf die Werte T an und erzeugt die Werte S des feinen Netzes
2:     Finde unter allen Zellen i aus dem feinen Netz die maximale x-Koordinate max_x
3:     for jede Zelle i do
4:         if z-Koordinate von i=max_z then i in list_max_z aufnehmen
5:         end if
6:         S[i] ungültigen Wert zuordnen
7:     end for
8:     Finde die Zelle i in list_max_z mit min. z-Koord. und setze start=i
9:     Finde die Zelle i in list_max_z mit max. z-Koord. und setze stop=i
10:    next=start
11:    for jede Zelle i am Rand do
12: findet alle Elemente am Rand auf der letzten Seite (nötig, da es bei den höhren Agglomerationen zu Verschiebungen kommt und die neuen Elemente nicht mehr exakt auf einer Linie liegen)
13:        if next=stop then for-Schleife verlassen
14:        end if
15:        if Existiert ein Nachbar i von next in list_max_z then next=i
16:        elsenext=Nachbar von next mit maximaler z-Koordinate
17:        end if
18:    end for
19:    for jede Zelle i do
20:        prolong_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse, start, v_1, v_2, v_3)
21:        for jede Nachbarzelle j der Zelle start do
22:            if coarse[start]=coarse[j] then
23:                prolong_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse, j, v_2, v_4, v_6)
24:                j in list_neigh aufnehmen
25:            end if
26:        end for
27:        for jede Zelle j in list_neigh do
28:            jede Nachbarzelle k der Zelle j mit S[k] ungültig in list_neigh_2 aufnehmen
29:        end for
30:        doppelt auftretende Zelle k in list_neigh_2 finden und prolong_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse, k, v_1, v_2, v_3) aufrufen
```

```
31:         for alle Zellen l in list_neigh_2 außer k do
32:             if start ist aus list_max_x then Zelle mit minimaler x-Koordinate finden und next=l
33:             else Zelle mit genau 2 Nachbarn m mit S[m] ungültig finden und next=l setzen
34:             end if
35:         end for
36:         if S[stop] nicht mehr ungültig then //neue Schicht
37:             Finde die Zelle l in list_max_z mit minimaler z-Koordinate und S[l] ungültig und setze start=l; Finde die Zelle l in list_max_z mit maximaler z-Koordinate und S[l] ungültig und setze stop=l
38:         elsestart=next
39:         end if
40:         if S[start] gültiger Wert then For-Schleife beenden
41:         end if
42:     end for
43:     return S
44: end procedure
```

Algorithm 5 Hilfsfunktion zur Restriktion

```
1: procedure RESTRIKT_HELP(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, S, coarse, zelle, v_i, v_j, v_l)
2:     T[coarse[zelle]]=T[coarse[zelle]]+v_i*S[zelle]
3:     for jede Nachbarzelle j der Zelle zelle do
4:         if coarse[j]≠ coarse[zelle] then
5:             T[coarse[zelle]]=T[coarse[zelle]]+v_j*S[j]
6:             j in list_neigh_help aufnehmen
7:         end if
8:     end for
9:     for jede Zelle j in list_neigh_help do
10:        jede Nachbarzelle k der Zelle j in list_neigh_2_help aufnehmen
11:    end for
12:    doppelt auftretende Zelle k mit k≠ zelle in list_neigh_2_help finden und T[coarse[zelle]]=T[coarse[zelle]]+v_l*S[k]
13:    return T[coarse[zelle]]
14: end procedure
```

Algorithm 6 Restriktion

```
1: procedure RESTRIKTION(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse)  ▷ Wendet den Prolongationsoperator auf die Werte T an und erzeugt die Werte S des feinen Netzes
2:     Finde unter allen Zellen i aus dem feinen Netz die maximale x-Koordinate max_x
3:     for jede Zelle i do
4:         if z-Koordinate von i=max_z then i in list_max_z aufnehmen
5:         end if
6:         S[i] ungültigen Wert zuordnen
7:     end for
8:     Finde die Zelle i in list_max_z mit min. z-Koord. und setze start=i
9:     Finde die Zelle i in list_max_z mit max. z-Koord. und setze stop=i
10:    next=start
11:    for jede Zelle i am Rand do
12: findet alle Elemente am Rand auf der letzten Seite (nötig, da es bei den höhren Agglomerationen zu Verschiebungen kommt und die neuen Elemente nicht mehr exakt auf einer Linie liegen)
13:        if next=stop then for-Schleife verlassen
14:        end if
15:        if Existiert ein Nachbar i von next in list_max_z then next=i
16:        elsenext=Nachbar von next mit maximaler z-Koordinate
17:        end if
18:    end for
19:    for jede Zelle i do
20:        T[coarse[start]]=0
21:        restrikt_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, S, coarse, start, v_1, v_2, v_3)
22:        for jede Nachbarzelle j der Zelle start do
23:            if coarse[start]=coarse[j] then
24:                restrikt_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, S, coarse, j, v_2, v_4, v_6)
25:                j in list_neigh aufnehmen
26:            end if
27:        end for
28:        for jede Zelle j in list_neigh do
29:            jede Nachbarzelle k der Zelle j mit T[coarse[k]] ungültig in list_neigh_2 aufnehmen
30:        end for
31:        vierte Zelle k, welche noch nicht zugeordnet wurde, aus coarse[start] finden und prolong_help(Netzstruktur_fein, Netzstruktur_grob, Neigh_id, Neighbors, T, coarse, k, v_1, v_2, v_3) aufrufen
```

```
32:         for alle Zellen l in list_neigh_2 außer k do
33:             if start ist aus list_max_x then Zelle mit minimaler x-Koordinate finden und next=l
34:             else Zelle mit genau 2 Nachbarn m mit T[coarse[m]] ungültig finden und next=l setzen
35:             end if
36:         end for
37:         if S[stop] nicht mehr ungültig then //neue Schicht
38:             Finde die Zelle l in list_max_z mit minimaler z-Koordinate und T[coarse[l]] ungültig und setze start=l; Finde die Zelle l in list_max_z mit maximaler z-Koordinate und T[coarse[l]] ungültig und setze stop=l
39:         elsestart=next
40:         end if
41:         if S[start] gültiger Wert then For-Schleife beenden
42:         end if
43:     end for
44:     return S
45: end procedure
```

DLR-IB-AS-BS-2020-20

Mehrgitteruntersuchungen zur Lösung der Navier-Stokes-Gleichungen

Sophie Pientka

Verteiler:

Institutsbibliothek	1 Exemplar
Verfasser	5 Exemplare
Institutsleitung	1 Exemplar
Abteilungsleiter	1 Exemplar
Deutsche Bibliothek in Frankfurt/Main	2 Exemplare
Niedersächsische Landesbibliothek Hannover	1 Exemplar
Techn. Informationsbibliothek Hannover	1 Exemplar
Zentralbibliothek BS	2 Exemplare
Zentralarchiv GÖ	1 Exemplar
Reserve	5 Exemplare